2019

# Metody programowania sterowników PLC

## WYKORZYSTANIE PRZEMYSŁOWYCH URZĄDZEŃ MIKROPROCESOROWYCH W STEROWANIU PROSTYMI I ZŁOŻONYMI PROCESAMI MECHATRONICZNYMI

DR INŻ. ZBIGNIEW SETA

UKŁAD CYFROWY, MIKROPROCESOR, STEROWNIK CYFROWY, STEROWNIK PROGRAMOWALNY PLC, SYSTEM MECHATRONICZNY, PROGRAMOWANIE URZĄDZEŃ MIKROPROCESOROWYCH, MODUŁY STEROWNIKA PLC, KONFIGURACJA STEROWNIKA PLC, NORMA IEC 1131, METODY PROGRAMOWANIA STEROWNIKÓW PLC, APLIKACJA STEROWNIKA PLC W SYSTEMACH MECHAT.

W niniejszej publikacji podjęto się zadania przedstawienia zagadnień stosowalności w sterowaniu mechatronicznym nowoczesnych mikroprocesorowych urządzeń – sterowników programowalnych PLC (ang. Programmable Logic Controllers). Procesy sterowania, na których skupiono się przy wyjaśnieniu zasad wykorzystania tych urządzeń obejmują przypadki prostych oraz złożonych układów mechatronicznych. Do tych pierwszych zaliczymy sterowanie siłownikami pneumatycznymi, zaś do tych drugich sterowanie napędami elektrycznymi z wykorzystaniem przetwornic częstotliwości. Za wszelkie uwagi dotyczące prezentowanego materiału Autor niniejszej publikacji będzie bardzo wdzięczny. Ewentualne uwagi Czytelnik może kierować pod adres e-mail: seta@mchtr.pw.edu.pl.

# Spis treści

# PRZEDMOWA

W niniejszym opracowaniu autor podjął się zadania przedstawienia stosowalności w sterowaniu systemami mechatronicznymi nowoczesnych mikroprocesorowych urządzeń procesu produkcyjnego – **Sterowników Programowalnych PLC** (ang. *Programmable Logic Controllers* - w skrócie: sterowników PLC). Ponieważ z racji swoich parametrów technicznych oraz funkcjonalnych sterownik PLC wykorzystywany jest w przeważającej większości przypadków w sterowaniu systemami mechatronicznymi, w których układ sterowania operuje na sygnałach wejścia/wyjścia, będących sygnałami dyskretnymi (dwustanowymi, pochodzącymi ze zbioru {0,1}), przy omawianiu zagadnień wykorzystania sterownika PLC w takim sterowaniu skupiono się przede wszystkim na omówieniu przykładów sterowania procesami dyskretnymi.

Przykłady sterowania systemami mechatronicznymi, które posłużyły autorowi do wyjaśnienia roli oraz znaczenia wykorzystania w takim systemie sterownika PLC, objęły przypadki prostych jak i złożonych systemów mechatronicznych. Do tych pierwszych układów zaliczono np. sterowanie silnikiem elektrycznym w różnych konfiguracjach. Do tych drugich układów zaliczono np. sterowanie pracą szybowej windy towarowej oraz procesem mieszania materiałów sypkich.

Przy prezentowaniu materiału w zakresie aplikacji fizycznej sterowników PLC do sterowania przykładowymi systemami mechatronicznymi skoncentrowano się na zilustrowaniu Czytelnikowi najważniejszych aspektów wykorzystania tych urządzeń w takim sterowaniu, czyli:

- wkomponowaniu sterownika PLC w układ sterowania przykładowym systemem mechatronicznym;
- syntezie algorytmu procesu metodą modelowania określaną jako GRAFCET;
- syntezie algorytmu sterowania metodą modelowania określaną jako SFC;
- tworzeniu zapisu programu wykonywalnego (użytkowego) dla sterownika PLC przy użyciu metod programowania, określonych w normie IEC 131-3;
- podaniu uwarunkowań związanych z bezpieczeństwem funkcjonowania sterownika PLC w układzie sterowania przykładowymi systemami mechatronicznymi.

W związku z faktem, iż w systemach mechatronicznych funkcjonują nadal układy, określane mianem układów sterowania stykowego, (czyli układy, w których dany algorytm sterowania realizowany jest za pomocą sterowania przekaźnikami, stycznikami, przyciskami, itp. i ich łącznikami stykowymi - zestykami), wszędzie tam, gdzie było to konieczne i wskazane, sterowanie za pomocą sterownika PLC odniesiono do takiego właśnie układu stykowego. Uczyniono tak dla lepszego zrozumienia przez Czytelnika prezentowanego materiału zakładając, że w niektórych przypadkach pokazanie analogii dwóch rodzajów sterowań, które spotykane są w mechatronice, odniesionych do tego samego zadania sterowania systemem mechatronicznym może być korzystne dla Czytelnika. Założono przy tym, że Czytelnik dysponuje podstawową wiedzą na temat działania oraz konstrukcji układów sterowania stykowego.

Autor zaznacza, że zakres materiału, zilustrowany w niniejszej publikacji podzielono na cztery następujące moduły:

⇒ ***Wykorzystanie mikroprocesorów w sterowaniu. Podstawy sterowników programowalnych PLC*** – treść tego modułu ilustruje najważniejsze cechy mikroprocesora, który jest „sercem" każdego sterownika PLC, bez względu na jego rodzaj czy typ, oraz ilustruje sposób włączenia (wkomponowania) sterownika PLC do systemu sterowania mechatronicznego wraz z przedstawieniem podstaw syntezy algorytmu sterowania, która powinna być dokonana przed utworzeniem programu sterującego dla sterownika PLC;

⇒ ***Budowa sterowników programowalnych PLC. Podstawowe moduły sterownika PLC*** – treść tego modułu omawia rodzaje sterowników PLC z podziałem na urządzenia typu Compact oraz typu modułowego, oraz prezentuje rodzaje oraz parametry techniczne modułów, które wchodzą w skład konfiguracji każdego sterownika PLC;

⇒ ***Norma przemysłowa IEC 1131-3. Metody programowania sterowników PLC*** – treść tego modułu omawia trzecią część normy (europejska sygnatura to EN 61131-3), w której zawarte jest omówienie metod programowania sterowników PLC;

⇒ ***Uruchamianie oraz testowanie programu w sterowniku PLC*** – treść tego modułu omawia włączenie sterownika PLC jako serca układu sterowania do przykładowych prostych oraz złożonych systemów mechatronicznych, włączając zilustrowanie uruchomienia sterownika PLC oraz późniejsze testowanie programu użytkowego na sterowanym obiekcie.

Autor zaznacza również, że niniejsza publikacja uzupełniona została odpowiednio dobranymi materiałami multimedialnymi, w dużej ich liczbie, które są swoistego rodzaju mini-wykładami autora opracowania, i które będą dostępne dla Czytelnika.

Dodatkowo autor opracowania przygotował dla każdego modułu zestaw pytań kontrolnych oraz testów sprawdzających wraz z odpowiedziami. Powyższe powinno dać Czytelnikowi odpowiedź, w jakim stopniu przyswoił on sobie materiał niniejszej publikacji.

Na zakończenie autor podkreśli, że niniejszą publikację opracowano na podstawie materiału, będącego treścią licznych wykładów, jakie autor prowadził dla studentów uczelni technicznych. Odbiorcami niniejszej publikacji powinni być Czytelnicy zajmujący się zawodowo projektowaniem aplikacji sterujących w systemach mechatronicznych, które zawierają m.in. urządzenia mikroprocesorowe typu sterowniki PLC. Publikacja będzie również przydatna dla studentów wydziałów elektrycznych, informatycznych, mechatronicznych, itp., uczelni technicznych, czyli wszędzie tam, gdzie występuje kształcenie studentów o specjalnościach pokrewnych automatyce, mechatronice oraz sterowaniu procesami technologicznymi.

Za wszelkie uwagi dotyczące prezentowanego materiału autor będzie bardzo wdzięczny. Czytelnicy mogą je kierować pod adres e-mail: seta@mchtr.pw.edu.pl lub zbigniew.seta@pw.edu.pl.

## Norma IEC 1131.

Warunkiem poprawnego spełnienia przez sterownik PLC swojej roli, czyli realizacji zadania sterowania dla kontrolowania szeroko rozumianych procesów mechatronicznych, jest uprzednie utworzenie programu sterującego dla tego urządzenia, który jest następnie umieszczany w jego pamięci programu. Utworzenie zaś takiego programu sterującego powinno być poprzedzone opracowaniem najpierw algorytmu procesu oraz później algorytmu sterowania przy użyciu odpowiednich formalizmów graficznych. Podstawowe zasady tych formalizmów opisano już wcześniej w p.1.6.3 (dla algorytmu GRAFCET) oraz p.1.6.4 (dla algorytmu SFC) niniejszej publikacji.

Przyjmuje się, że modelowanie algorytmu procesu mechatronicznego jako takiego odbywa się przy użyciu algorytmu **GRAFCET**, zaś algorytm sterowania modeluje się za pomocą struktur graficznych **SFC**, co jest później bardzo pomocne przy tworzeniu programu użytkowego dla sterownika PLC, ponieważ cechą modelowania graficznego **SFC** jest używanie w tym grafie instrukcji oraz tzw. warunków przejścia w sposób podobny do takiego, jaki występuje w językach (metodach) programowania sterowników PLC.

Pojawienie się i rozwój sterowników PLC prowadził do systematycznych prób formalizowania zasad jego budowy logicznej jako urządzenia cyfrowego, wraz z określeniem sposobu komunikacji pomiędzy jego blokami funkcyjnymi oraz do prób formalizowania metod tworzenia programu użytkowego, czyli w istocie do sformalizowania metod (języków) programowania tych urządzeń dla systemów mechatronicznych. Zaowocowało to w konsekwencji opublikowaniem w 1993 r. normy IEC 1131 (EN 61131) po tytułem: „***Programmable Controllers***", której ustalenia zawarto w pięciu rozdziałach - arkuszach (w momencie opublikowania normy opracowane były tylko trzy pierwsze arkusze):

- Informacje ogólne;
- Wymagania sprzętowe i testy;
- Języki (metody) programowania;
- Wskazówki dla użytkowników;
- Specyfikacja przesyłania wiadomości w obrębie systemu PLC.

Zatem norma IEC 1131 określiła m.in. standardy w programowaniu sterowników PLC oraz podała formalizm graficzny dla budowy algorytmu SFC. Algorytm ten w owym czasie, czyli w momencie opublikowania normy zaliczany był do grupy formalizmów wyłącznie dla graficznego modelowania algorytmu sterowania systemem mechatronicznym, zaś obecnie modelowanie SFC uważane jest już za jedną z metod graficznych tworzenia programu użytkowego dla sterownika PLC. Omówienie wzmiankowanej normy ze szczególnym naciskiem położonym na trzeci jej arkusz - języki programowania, czyli arkusz IEC 1131-3 stanowi materiał trzeciego modułu tej książki. Prezentując zalecenia tej normy w tym zakresie, gdy było to niezbędne, zaprezentowano je na tle konstrukcji programu dla sterownika PLC w wybranym oprogramowaniu narzędziowym – oprogramowaniu STEP 7 dla sterowników PLC firmy Siemens.

## 1. Informacje ogólne (arkusz IEC 1131-1)

W pierwszym arkuszu normy IEC 1131 podano m.in. ogólne definicje, które dotyczą nazewnictwa oraz pojęć mających zastosowanie zarówno w sprzęcie jak i oprogramowaniu, związanym ze sterownikami PLC, oraz określono sposób identyfikacji głównych cech, które są istotne przy wyborze sterownika PLC do konkretnej aplikacji sterowania, jak również do wyboru urządzeń peryferyjnych, które mogą współpracować ze sterownikiem PLC. Podanie tych informacji miało na celu wyeliminowanie wszelkich nieścisłości językowych, mogących powodować nieporozumienia wśród szerokiej grupy (czytaj: międzynarodowej) użytkowników sterowników PLC. Pierwszy arkusz normy IEC 1131 wprowadza m.in. następujące poniższe pojęcia:

⇒ ***PROGRAM UŻYTKOWY PLC*** lub ***PROGRAM UŻYTKOWNIKA*** - logiczne złożenie wszystkich elementów języka programowania i konstrukcji logicznych, które konieczne są dla zamierzonego przetwarzania sygnałów wejścia/wyjścia systemu mechatronicznego, wymaganych do sterowania maszyną lub procesem;

⇒ ***SYSTEM ZAUTOMATYZOWANY*** - system kontroli, który wykracza poza zakres objęty normą IEC 1131, ale który może zawierać również i inne składniki, objęte ustaleniami technicznymi konkretnych norm;

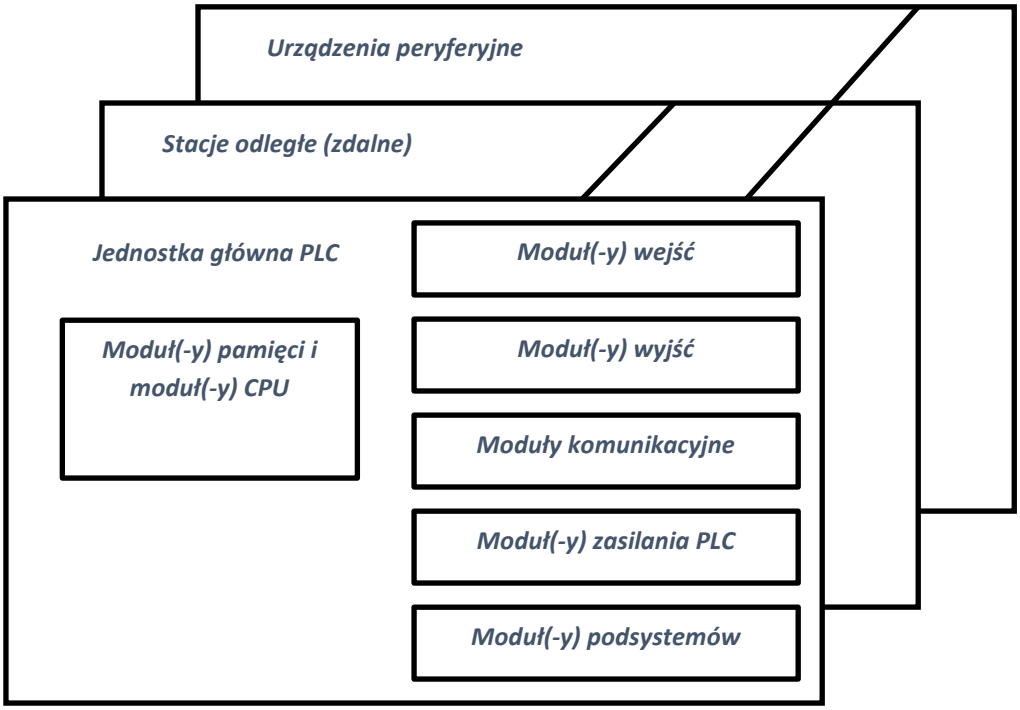
⇒ ***URZĄDZENIE PERYFERYJNE*** – urządzenie lub moduł, wyposażony w interfejs wejścia/wyjścia, zapewniający odbiór oraz przetwarzanie danych, które zostaną przesłane do programu użytkowego sterownika PLC. Urządzeniem peryferyjnym może być urządzenie zewnętrzne, zainstalowane w systemie mechatronicznym (tzw. obiektowe), które działa niezależnie od sterownika PLC i nie jest przez program użytkowy synchronizowane, lub może to być moduł rozszerzeniowy sterownika PLC, który podłączony jest do sterownika PLC za pomocą magistrali sygnałowej.

⇒ ***SCHEMAT DRABINKOWY*** - jedna lub kilka pionowo rozmieszczanych (jedna pod drugą) tzw. sieci zestyków, cewek, funkcji, itp., przedstawianych graficznie za pomocą odpowiednio połączonych symboli i bloków funkcyjnych. Sieci te ograniczane są umyślonym miejscem niższego potencjału, który sugerowany jest po lewej stronie arkusza schematu i (opcjonalnie) ograniczane są miejscem wyższego potencjału, który sugerowany jest po prawej stronie arkusza schematu;

⇒ ***STEROWNIK CYFROWY PLC*** - działający system elektroniczny, zaprojektowany do użytku w środowisku przemysłowym, który wykorzystuje pamięć programowalną do wewnętrznego przechowywania zorientowanych na użytkownika instrukcji, używanych do implementacji określonych funkcji, takich jak instrukcje logiczne, sekwencjonowanie zdarzeń, odmierzanie czasu, liczenie, itp., czyli przechowywania programu użytkownika. Sterownik PLC poprzez „swoje" cyfrowe i analogowe wejścia/wyjścia kontroluje oraz steruje różnymi typami maszyn lub procesów systemu mechatronicznego. Zarówno sterownik PLC jak i związane z nim urządzenia peryferyjne są tak zaprojektowane, aby można było je łatwo zintegrować, tworząc przemysłowy system kontroli i sterowania, łatwy do użycia we wszystkich zamierzonych funkcjach aplikacji sterownika PLC;

⇒ ***PROGRAMOWALNY SYSTEM PLC*** - konfiguracja użytkownika, składająca się ze sterownika PLC i powiązanych urządzeń peryferyjnych, która jest niezbędna dla zamierzonego celu sterowania systemem mechatronicznym. Konfiguracja taka składa się z jednostek przewidzianych do współpracy w ramach sterownika PLC, które połączone są ze sobą na stałe przy pomocy przewodów (kabli) i połączeń wtykowych lub innych środków, np. transmisji bezprzewodowych;

Pierwszy arkusz normy IEC 1131 określił zatem, aby model sterownika PLC, uwzględniający jego konfigurację sprzętową cechował się postacią, która funkcjonalnie pokazana została na rysunku 38.




*Rysunek 38: Postać blokowa konfiguracji sprzętowej systemu PLC według normy IEC 1131-1*

W zakresie spełnianych zadań, które realizują poszczególne części logiczne sterownika PLC, pierwszy arkusz normy IEC 1131 nakreślił następujące funkcje:

1. funkcja przetwarzania sygnałów;
2. funkcja interfejsu z czujnikami i elementami wykonawczymi;
3. funkcja komunikacji;
4. funkcja interfejsu człowiek - maszyna **MMI**;
5. funkcja programowania, uruchamiania, testowania i dokumentacji;
6. funkcja zasilania.

Wyszczególnioną strukturę zadaniową sterownika PLC pokazano na rysunku 39 zaś opisano poniżej.

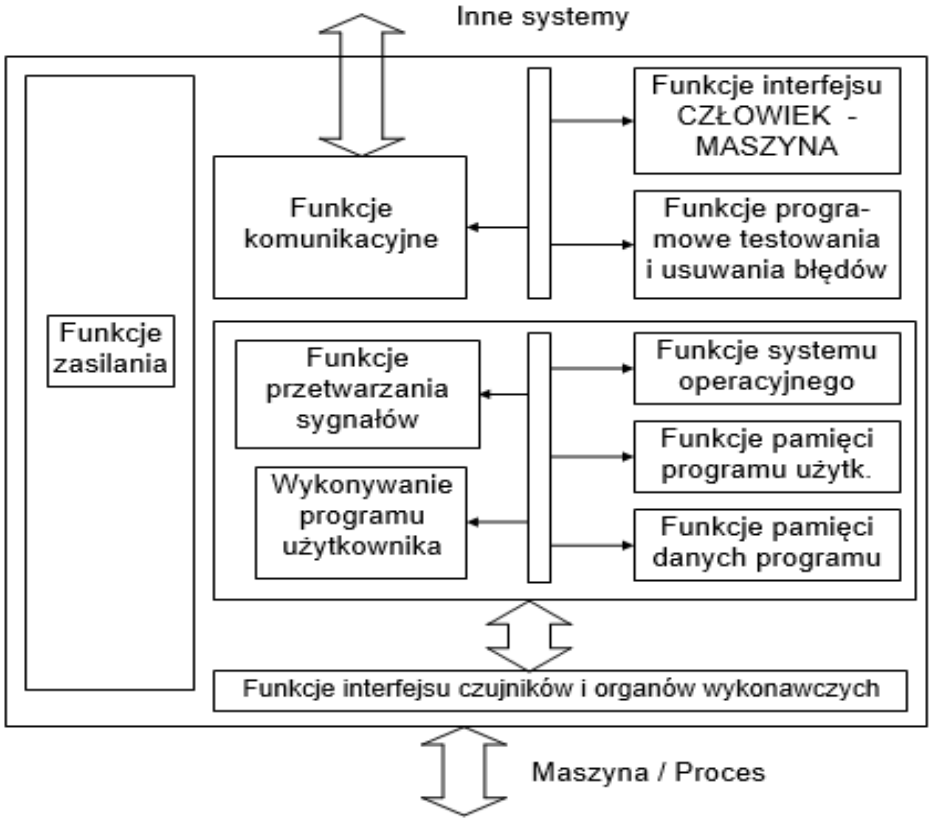
### 1.1. Funkcja przetwarzania sygnałów

Podzespół realizujący funkcje przetwarzania sygnałów zawiera pamięć programu użytkowego, pamięć danych, system operacyjny sterownika PLC oraz funkcje wykonywania programu użytkowego. Podzespół ten przetwarza sygnały pochodzące z czujników procesowych (moduły wejść sterownika) lub pamięci sterownika i wypracowuje

sygnały do elementów wykonawczych systemu mechatronicznego (moduły wyjść sterownika), a także przekazuje dane do pamięci danych zgodnie z programem użytkowym.

Możliwości sterownika PLC określone są przez następujące funkcje:

- sterowanie logiczne (bloki logiczne, przekaźniki czasowe, liczniki itp.);
- sterowanie sekwencyjne;
- przetwarzanie sygnałów lub danych (funkcje matematyczne, obsługa danych, przetwarzanie danych analogowych);
- funkcje sprzęgania (urządzenia we/wy, inne systemy, drukarki, pamięć masowa itp.);
- sterowanie wykonywaniem programu użytkowego;
- konfiguracja systemu sterownika PLC.




*Rysunek 39: Struktura funkcjonalna sterownika PLC według normy IEC 1131-1*

Według cytowanej normy zadaniem systemu operacyjnego sterownika PLC jest zarządzanie wewnętrznymi, wzajemnie od siebie zależnymi funkcjami systemu PLC takimi jak: kontrola konfiguracji sterownika, diagnostyka programu użytkowego i wykrywanie błędów, zarządzanie pamięcią, itp. Po wyłączeniu sterownika PLC i jego ponownym załączeniu urządzenie powinno rozpocząć ponowną pracę na trzy niżej wymienione sposoby:

1. **Zimny restart** - restart sterownika PLC i wszystkich zaimplementowanych w pamięci programu aplikacji po odświeżeniu wszystkich danych dynamicznych (np. zmiennych programowych, rejestrów dla modułów wejść/wyjść, rejestrów wewnętrznych, zegarów, liczników, itp.), które są resetowane do wcześniej określonego stanu, który występował przed zimnym restartem. Zimny restart może być przeprowadzony automatycznie przez system operacyjny sterownika PLC, np. po wcześniejszej utracie i powrocie zasilania urządzenia, lub może być wymuszony ręcznie, np. poprzez użycie odpowiedniego przycisku, który znajduje się na płycie czołowej sterownika PLC (zazwyczaj na płycie czołowej modułu CPU).

2. **Ciepły restart** – restart sterownika PLC po przywróceniu napięcia zasilania, po wykonaniu którego program użytkownika realizowany jest od początku. Obszary pamięci zadeklarowane jako nie podtrzymywane zostają wyzerowane, zaś obszary pamięci podtrzymywane przyjmują wartości jak przed wyłączeniem sterownika.
3. **Gorący restart** – restart sterownika PLC, który występuje po krótkotrwałym zaniku napięcia zasilania. Po gorącym restarcie sterownik PLC wraca do normalnego stanu sprzed wystąpienia awarii. Wszystkie dane, które pochodzą z rejestrów wejścia/wyjścia oraz dane dynamiczne są odzyskane lub nie są zmienione. Funkcja gorącego restartu wymaga osobnego zasilania zegara czasu rzeczywistego lub timera do określenia czasu, jaki upłynął od wykrycia zaniku zasilania sterownika PLC.

Zadaniem pamięci programu użytkowego jest zapewnienie miejsca do przechowywania serii rozkazów, które określają działanie sterownika jako całości, a pamięć danych użytkowych powinna zapewniać miejsce do przechowywania bloku danych wejściowych i wyjściowych oraz danych, które niezbędne są podczas wykonywania programu użytkownika. Norma IEC 1131-1 dopuszcza wykorzystanie w sterowniku PLC następujących rodzajów pamięci:

- **RAM** – pamięci do zapisu i odczytu z utratą danych przy braku zasilania pamięci; pamięć zachowuje dane przy podtrzymaniu bateryjnym jej zasilania;
- **ROM** – pamięć wstępnie zaprogramowana tylko do odczytu z zachowaniem danych przy jej braku zasilania;
- **PROM** – pamięć programowalna tylko do odczytu z zachowaniem danych przy jej braku zasilania;
- **EPROM/UV-PROM**, **EEPROM** – pamięć z możliwością wielokrotnego kasowania w specjalnych kasownikach/programatorach oraz zaprogramowania; tylko do odczytu z zachowaniem danych przy braku zasilania pamięci;

Utrzymanie niezbędnych danych na wypadek utraty napięcia zasilania sterownika PLC powinno być zagwarantowane poprzez wybór odpowiedniego rodzaju pamięci, która zachowuje dane przy takiej awarii (np. pamięć EPROM/EEPROM) lub poprzez zastosowanie rozwiązania podtrzymania napięcia zasilania pamięci za pośrednictwem zespołu baterii.

Norma IEC 1131-1 określiła wyrażanie pojemności pamięci (dla programu użytkowego i danych) w kilobajtach (1KB= 1024 bity) i narzuciła odniesienie jej wielkości do:

- pojemności pamięci dla minimalnej konfiguracji sprzętowej sterownika PLC;
- pojemności pamięci dla ilości modułów rozszerzających, które wchodzą w skład konfiguracji sterownika PLC;
- pojemności pamięci dla maksymalnej konfiguracji sterownika PLC, czyli dla ilości modułów we/wy, które mogą wchodzić w skład konfiguracji sprzętowej sterownika PLC.

Program użytkowy może składać się z wielu zadań (ang. ***Tasks***). Wykonanie każdego zadania jest realizowane sekwencyjnie wiersz po wierszu („drabinka” po drabince), aż do końca całego zadania, co kontroluje system operacyjny sterownika PLC. Przy wykryciu określonego zdarzenia (np. przerwania zewnętrznego, które posiada wyższy priorytet niż wykonywane bieżące zadanie), następuje warunkowe zatrzymanie realizacji bieżącego zadania na rzecz rozpoczęcia obsługi programowej przerwania zewnętrznego aż do końca jego realizacji. Gdy to nastąpi kontynuowana jest realizacja bieżącego zadania (tego przed wystąpieniem przerwania zewnętrznego).

### 1.2. Funkcja interfejsu z czujnikami i elementami wykonawczymi

Zadaniem podzespołu, który realizuje funkcję interfejsu z czujnikami i elementami wykonawczymi jest z jednej strony przekazywanie informacji o stanie danych systemu mechatronicznego, kontrolowanego przez sterownik PLC, a z drugiej strony wysyłanie sygnałów, które uruchamiają elementy wykonawcze tegoż systemu.

Norma IEC 1131-1 narzuca, aby sygnały z czujników systemu mechatronicznego były dostarczane do modułu wejść sterownika PLC, zaś sygnały, przeznaczone dla elementów wykonawczych były osiągane w modułach wyjść sterownika PLC. Norma IEC 1131-1 narzuca również, aby do przenoszenia tych informacji w obrębie modułów wejść/wyjść sterownika PLC posługiwano się sygnałami binarnymi (cyfrowymi) oraz sygnałami analogowymi.

Norma IEC 1131-1 narzuca, aby w modułach wejść/wyjść sterownika PLC stosowano różne metody przetwarzania, konwersji oraz separacji sygnałów, oraz że systemy sterowników PLC muszą charakteryzować się modularnością, pozwalającą na stosowanie konfiguracji sterownika PLC zgodnie z potrzebami aplikacji sterowania systemami mechatronicznymi.

### 1.3. Funkcja komunikacji

Funkcja komunikacji, ujęta w normie IEC 1131-1 reprezentuje właściwości komunikacyjne sterowników PLC do nawiązania połączenia w obrębie systemu mechatronicznego. Funkcja komunikacji obsługuje program użytkowy oraz wymianę danych między sterownikiem PLC a urządzeniami zewnętrznymi (peryferyjnymi), wymianę danych z innymi sterownikami PLC lub dowolnymi urządzeniami systemu mechatronicznego. Funkcja komunikacji powinna zapewnić m.in. weryfikację urządzeń, z którymi sterownik PLC nawiązuje połączenie oraz powinna zapewnić również gromadzenie danych, niezbędnych dla programu użytkowego i raportowanie alarmów, jako efekt ewentualnych nieprawidłowości w pracy sterownika PLC. Funkcja komunikacji realizowana jest zazwyczaj za pośrednictwem szeregowej transmisji danych, która dominuje w sieciach lokalnych typu **LAN** (ang. ***L**ocal **A**rea **N**etwork*) lub łącza typu punkt – punkt **P2P** (ang. ***P**eer to **P**eer*).

### 1.4. Funkcja interfejsu człowiek - maszyna MMI

Na funkcję interfejsu człowiek - maszyna **MMI** (ang. ***H**uman **M**achine **I**nterface*) norma IEC 1131-1 narzuciła dwa cele:

- dostarczanie operatorowi systemu mechatronicznego informacji niezbędnych do monitorowania działania procesu sterowania z wykorzystaniem sterownika PLC;
- umożliwianie operatorowi systemu mechatronicznego oddziaływania na system sterownika PLC i jego program(-y) użytkowy(-e) w celu podejmowania decyzji odnośnie sterowania systemem mechatronicznym.

### 1.5. Funkcja programowania, uruchamiania, testowania i dokumentacji

Norma IEC 1131-1 narzuciła, aby program użytkowy sterownika PLC był wprowadzany do tego urządzenia za pomocą urządzeń wyposażonych w klawiaturę alfanumeryczną lub symboliczną, co powinno odbywać się za pośrednictwem klawiszy kursorów (tzw. dżojstik) lub za pomocą myszy komputerowej. Wszystkie wprowadzane programy do sterownika PLC powinny być sprawdzane pod względem poprawności i wewnętrznej zgodności.

Podczas tworzenia programu wszystkie rozkazy muszą być wyświetlane w urządzeniu programującym lub na ekranie monitora (komputer PC, laptop, itp.) oraz powinna istnieć możliwość dokumentacji programu. Dodatkowo, podczas uruchamiania programu użytkowego powinny być spełnione warunki bezpośredniej komunikacji (ang. *On - line*) z urządzeniem programującym.

Funkcjami testującymi, wspomagającymi użytkownika podczas pisania, uruchamiania i testowania programu są:

- sprawdzanie stanów w rejestrach modułów wejść/wyjść oraz funkcji wewnętrznych (timery, liczniki, itp.);
- sprawdzanie sekwencji programu, np.: praca krokowa;
- symulacja funkcji interfejsów, np.: wymuszanie stanów w rejestrach wejść/wyjść.

Norma IEC 1131-1 zaleca, aby dokumentacja systemu PLC dostarczała pełnego opisu zastosowanego sterownika PLC i jego infrastruktury elektrycznej, czyli powinna zawierać m.in.:

- opis konfiguracji sprzętu;
- dokumentację programu użytkownika zawierającą m in.: wydruk programu użytkowego, komentarze, opis modyfikacji itp.;
- dokumentację serwisową.

Dla tworzenia programu dla sterownika PLC norma IEC 1131-1 przewidziała użycie następujących języków (metod) programowania:

- język **IL** (ang. ***I****nstruction* ***L****ist*) lub **STL** (ang. ***S****tatement* ***L****ist*);
- język **ST** (ang. ***S****tructuret* ***T****ext*);
- język **FBD** (ang. ***F****unction* ***B****lock* ***D****iagram*);
- język **LAD** (ang. ***L****adder* ***D****iagram*);
- metoda SFC (ang. ***S****equential* ***F****unction* ***T****ext*).

Norma IEC 1131-1 narzuca, aby wygenerowany program użytkowy znajdował się w pamięci programu (nieulotnej) sterownika PLC lub w pamięci typu **PADT** (ang. ***P****rogramming* ***A****nd* ***D****ebugging* ***T****ools*). To ostatnie wymaga wcześniejszego przeniesienia programu dla sterownika PLC do PADT i włożenie jej do odpowiedniego slota (najczęściej gniazda w module CPU) przed uruchomieniem sterownika PLC. (Niejednokrotnie w pamięci typu PADT przechowuje się prosty poprawny program, który jest wtedy inicjowany przez system operacyjny sterownika PLC jako pierwszy, co pomaga przy zawieszeniu się w działaniu tego urządzenia). Funkcje modyfikacji programu użytkownika powinny umożliwiać zmianę, dostosowanie i korektę programu. Typowymi funkcjami są: wyszukiwanie, zamiana, wstawianie, usuwanie i dodawanie znaków.

Według cytowanej normy możliwość przekazywania informacji o kontrolowanym systemie mechatronicznym oraz statusie wewnętrznym systemu, opartym o sterownik PLC ułatwia uruchomienie i debugowanie aplikacji PLC. Typowymi środki służącymi temu celowi to:

- wskazywanie statusu bezpośrednich wejść oraz wyjść sterownika PLC;
- wskazywanie lub/i rejestracja zmian statusu sygnałów zewnętrznych i danych wewnętrznych;
- monitorowanie czasu skanowania sygnałów wejściowych oraz czasu wykonania instrukcji programowych;

- wizualizacja w czasie rzeczywistym wykonania programu i przetwarzania danych;
- wskaźniki stanu ochrony zabezpieczeń elektrycznych układu sterownika PLC.

W celu szybkiej naprawy i zminimalizowania przestojów powinno być możliwe automatyczne zapisywanie programu PLC na nieulotnych nośnikach pamięci takich jak: Flash, karty PC, EEPROM, EPROM, dyski twarde, itp. Taki zapis powinien być aktualizowany po każdej modyfikacji programu tak, aby wystąpiła identyczność programu realizowanego aktualnie w sterowniku PLC z tym, zarchiwizowanym na nieulotnym nośniku pamięci.

### 1.6. Funkcja zasilania

Według normy IEC 1131-1 funkcja zasilania generuje napięcia niezbędne do obsługi systemu mechatronicznego, opartego o sterownik PLC, czyli zasilania nie tylko tego urządzenia, ale również i innych, współpracujących urządzeń. Funkcja zasilania zapewnia również sygnały sterujące dla właściwej synchronizacji ON/OFF urządzeń systemu PLC. Funkcja zasilania powinna umożliwiać dostarczenie różnych źródeł zasilania, co może być uwarunkowane zużyciem energii przez urządzenia systemu PLC lub spełnieniem wymagań odnośnie bezprzerwowego działania urządzeń systemu, itp.

Reasumując pierwsza część normy IEC 61131 narzuca, aby każdy system mechatroniczny, kontrolowany za pośrednictwem sterownika PLC, osiągał pewien poziom niezawodności oraz dostępności. Obowiązkiem użytkownika takiego systemu jest zagwarantowanie, aby architektura systemu mechatronicznego oraz charakterystyka samego systemu PLC i jego programu(-ów) użytkowego(-ych) spełniały zamierzone wymagania aplikacji sterowania takim systemem mechatronicznym.

## 2. Wymagania sprzętowe i testy (arkusz IEC 1131-2)

Druga część normy IEC 1131 zawiera następujące informacje:

- opis wymagań elektrycznych, mechanicznych i funkcjonalnych sterowników i ich urządzeń peryferyjnych;
- opis warunków użytkowania, przechowywania i transportu urządzeń;
- opis metod badań i procedury spełnienia wymagań w stosunku do sterowników PLC.

W części definicyjnej określone są m in. następujące terminy:

- współczynnik wykrywania nieobecności;
- odstęp izolacyjny powierzchniowy;
- obudowa;
- odporność;
- izolacja i inne.

## 3. Języki (metody) programowania (arkusz IEC 1131-3)

Arkusz trzeci normy IEC 1131 dotyczy pojęć podstawowych, zasad ogólnych, modelu programowego oraz modelu komunikacyjnego, który powinien być wykorzystywany przy wymianie danych między elementami oprogramowania. Arkusz ten określa również podstawowe typy i struktury danych, które powinny być używane przy tworzeniu programu sterującego dla sterownika PLC oraz wyszczególnia dwie grupy języków (metod) programowania sterownika PLC: języki (metody) tekstowe i języki (metody) graficzne.

W grupie języków tekstowych zdefiniowane zostały następujące języki:

- Język listy rozkazów **IL** (**STL**) (ang. ***I**nstruction **L**ist (**S**tatement **L**ist)*), porównywany często z językiem typu asembler; zbiór instrukcji w tym języku obejmuje m.in. operacje logiczne, arytmetyczne, relacji, jak również funkcje przerzutników, czasomierzy i liczników;
- Język tekstu strukturalnego **ST** (ang. ***S**tructured **T**ext*), będący odpowiednikiem języka wysokiego poziomu, zawierający struktury programowe i polecenia podobne do tych, które występują w takich językach jak PASCAL czy C++.

Do grupy języków graficznych opisanych w arkuszu trzecim normy IEC 1131 należą:

- Język „drabinkowy" **LD** (**LAD**) (ang. ***L**adder **D**iagram*), zbliżony konstrukcją graficzną do obwodów przekaźnikowych z zastosowaniem ich zestyków, w którym oprócz symboli zestyków, symboli cewek oraz połączeń między tymi elementami, dopuszcza się także użycie różnych funkcji (arytmetycznych, logicznych, porównań, relacji, itp.) oraz użycie bloków funkcjonalnych (przerzutników, czasomierzy, liczników, itp.). Wizualnie konstrukcja programu sterującego w metodzie **LD** odróżnia się od schematu stykowego tym, iż ten pierwszy schemat tworzony jest niejako od lewej strony do prawej (tzn. „zestyki" programu występują po lewej stronie, a „cewki" po prawej), zaś ten drugi z góry na dół według podobnych zasad analizy programu.
- Język funkcjonalny **FBD** (ang. ***F**unction **B**lock **D**iagram*), stanowiący odpowiednik schematu przepływu sygnałów, który spotykany jest w obwodów logicznych, przedstawianych w formie połączonych bramek logicznych typu AND, OR i NOT oraz funkcji i bloków funkcjonalnych, takich jak w języku **LD**.
- Graf (metoda) **SFC** (ang. ***S**equential **F**unction **C**hart*), stanowiący sposób tworzenia struktury wewnętrznej programu w postaci schematu funkcji sekwencyjnej, co pozwala na opisywanie zadań sterowania sekwencyjnego za pomocą grafów, zawierających kroki (tzw. etapy procesu) i warunki przejścia (tzw. tranzycje), czyli warunki logiczne niezbędne do „przemieszczania" się umyślnego sterowania od kroku poprzedniego do następnego.

Trzecia część normy IEC 1131 specyfikuje również syntaktykę i semantykę wymienionych języków programowania. Syntaktyka opisuje elementy języka i sposób ich użycia, natomiast semantyka ich znaczenie. Zdefiniowane zostały w normie również elementy konfiguracji, które wspomagają instalację oprogramowania w sterownikach PLC, oraz zdefiniowano wymagania komunikacyjne w celu ułatwienia połączenia sterowników PLC z innymi elementami systemu sterowania mechatronicznego.

### 3.1. Definicje mające zastosowanie w normie

Na potrzeby trzeciej części normy IEC 1131 wprowadzono następujące definicje m.in.:

- **Czas bezwzględny –** kombinacja czasu dnia oraz daty;
- **Ścieżka dostępu –** połączenie symbolicznej nazwy ze zmienną do celów komunikacji otwartej;
- **Akcja –** zmienna boolowska lub zestaw operacji, jakie należy wykonać wraz z odpowiednią strukturą sterowania;
- **Blok akcji –** element języka graficznego wykorzystujący boolowską zmienną wejścia w celu ustalenia wartości boolowskiej zmiennej wyjścia lub uaktywnienia warunków wystąpienia akcji, zgodnie ze strukturą sterowania określoną wcześniej;
- **Agregat –** uporządkowany zbiór obiektów danych, określony jako typ danych;
- **Argument -** synonim parametru wejścia lub wyjścia;
- **Przypisanie –** mechanizm nadawania wartości zmiennej;
- **Liczba o podstawie –** liczba, której reprezentacja ma określoną podstawę inną niż dziesięć;
- **Dwustanowy blok funkcji –** blok funkcji o dwu stanach stabilnych, sterowany przez jedno lub więcej wejść;
- **Ciąg bitów –** element danych składający się z jednego lub więcej bitów;
- **Ciało –** część jednostki organizacyjnej programu określająca operacje, które mają być wykonane na zadeklarowanych operandach jednostki organizacyjnej programu po rozpoczęciu jego wykonywania;
- **Wywołanie –** konstrukcja językowa rozpoczynająca wykonanie funkcji lub bloku funkcji;
- **Komentarz –** konstrukcja językowa pozwalająca włączyć do programu jakikolwiek tekst, nie wpływający na wykonanie programu;
- **Kompilować –** tłumaczyć jednostkę organizacyjną programu lub specyfikację typu danych na jej odpowiednik w języku maszyny lub w innej formie pośredniej;
- **Konfiguracja –** element językowy odpowiadający systemowi sterowników PLC, jak zdefiniowano w arkuszu pierwszym normy;
- **Blok funkcji licznika –** blok funkcji kumulujący wartość liczby zmian stwierdzonych na jednym lub więcej konkretnych wejść;
- **Typ danych –** zbiór wartości wraz z zestawem operacji, które mogą być wykonywane na tych wartościach danych;
- **Data i czas –** data w roku i czas dnia;
- **Deklaracja –** mechanizm tworzenia definicji elementu językowego. Deklaracja zwykle wymaga przypisania identyfikatora elementu językowego i określenia atrybutów takich jak typy danych i algorytmy tego elementu;
- **Ogranicznik –** znak lub kombinacja znaków służąca do oddzielenia elementów językowych programu;
- **Reprezentacja bezpośrednia –** sposób przedstawiania zmiennej w programie sterownika PLC, z którego można bezpośrednio odtworzyć określony przez producenta związek z położeniem logicznym lub fizycznym;
- **Słowo podwójne –** element danych składający się z 32 bitów;

- **Wartościowanie –** proces ustalania wartości wyrażenia, funkcji lub wyjść z sieci czy bloku funkcji podczas wykonywania programu;
- **Element sterowania wykonaniem –** element językowy sterujący przebiegiem wykonania programu;
- **Zbocze opadające –** przejście zmiennej boolowskiej z wartości „1” do wartości „0”;
- **Funkcja –** jednostka organizacyjna programu, która w trakcie wykonania dostarcza dokładnie jeden element danych (mogący mieć kilka wartości, np. tablica czy struktura), a także, której wywołanie można stosować w językach tekstowych jako operand w wyrażeniu;
- **Blok funkcji –** egzemplarz typu bloku funkcji;
- **Typ bloku funkcji –** element językowy programowania sterownika PLC składający się z:
  - definicji struktury danych rozdzielonej na zmienne wejścia, wyjścia i wewnętrzne;
  - zestawu operacji, które należy wykonać na elementach struktury danych w momencie wywołania egzemplarza bloku funkcji;
- **Schemat bloków funkcji –** jedna lub więcej sieci funkcji, bloków funkcji, elementów danych, etykiet i elementów połączenia, przedstawionych w formie graficznej;
- **Uniwersalny typ danych –** typ danych reprezentujący kilka typów danych;
- **Zakres globalny –** zakres deklaracji odnoszący się do wszystkich jednostek organizacyjnych programu, wewnątrz zasobu lub konfiguracji;
- **Zmienna globalna –** zmienna, której zakres jest globalny;
- **Adresowanie hierarchiczne –** reprezentacja bezpośrednia elementu danych jako części pewnej hierarchii fizycznej lub logicznej;
- **Identyfikator –** kombinacja liter, cyfr i znaków podkreślenia, rozpoczynająca się literą lub znakiem podkreślenia i nazywająca element językowy;
- **Wartość początkowa –** wartość przypisana zmiennej w momencie uruchomienia systemu;
- **Parametr wejścia (wejście)** - parametr służący do dostarczania argumentu jednostce organizacyjnej programu;
- **Egzemplarz –** pojedyncza, nazwana kopia struktury danych, związana z typem bloku funkcji lub typem programu, istniejąca między jednym a drugim wywołaniem powiązanych ze sobą operacji;
- **Nazwa egzemplarza –** identyfikator związany z konkretnym egzemplarzem;
- **Instancjacja –** tworzenie egzemplarza;
- **Literał całkowity –** literał reprezentujący bezpośrednio wartość typu SINT, INT, DINT, LINT, BOOL, BYTE, WORD, DWORD lub LWORD;
- **Rozpoczęcie –** proces inicjowania wykonania operacji określonych przez jednostkę organizacyjną programu;
- **Słowo kluczowe –** jednostka leksykalna charakteryzująca element językowy, np. „IF”;
- **Etykieta –** konstrukcja językowa nazywająca instrukcję, sieć lub grupę sieci oraz zawierająca identyfikator;
- **Zakres lokalny –** zakres deklaracji lub etykiety odnoszący się wyłącznie do jednostki organizacyjnej programu;

- **Położenie logiczne –** położenie zmiennej zaadresowanej w sposób hierarchiczny w schemacie, który może lub nie zawierać połączenia ze strukturą fizyczną wejść, wyjść i pamięci sterowników PLC;
- **Pamięć –** jednostka funkcyjna, w której program użytkownika może zapamiętać dane i skąd może pobierać zapamiętane dane;
- **Element nazwany –** element struktury oznaczony przez związany z nim identyfikator;
- **Blok funkcji opóźniający włączanie/wyłączanie –** blok funkcji opóźniający o określony czas zbocze opadające (narastające) wejścia boolowskiego;
- **Operand –** element językowy, na którym jest wykonywana operacja;
- **Operator –** symbol przedstawiający akcję, którą należy wykonać w trakcie operacji;
- **Parametr wyjścia (wyjście) –** parametr służący do odsyłania wyników wartościowania jednostki organizacyjnej programu;
- **Przepływ mocy –** symboliczne przedstawienie przepływu energii elektrycznej w schemacie drabinkowym, stosowane do wskazania postępu realizacji algorytmu logicznego;
- **Programować –** projektować, pisać i testować programy użytkowe;
- **Jednostka organizacyjna programu –** funkcja, blok funkcji lub program;
- **Zasób –** element językowy odpowiadający funkcji przetwarzania sygnału i jej interfejsowi człowiek – maszyna, jak również jej ewentualnym funkcjom interfejsu czujników i elementów wykonawczych, zdefiniowanych w pierwszej części normy IEC 1131;
- **Dane podtrzymywane –** dane zapamiętane w taki sposób, że ich wartość pozostaje niezmienna po wyłączeniu i ponownym włączeniu zasilania sterownika PLC;
- **Powrót –** element językowy w jednostce organizacyjnej programu oznaczający koniec sekwencji wykonywanych w tej jednostce;
- **Zbocze narastające –** przejście zmiennej boolowskiej z wartości „0” na „1”;
- **Zakres –** część elementu językowego, wewnątrz którego stosuje się deklarację lub etykietę;
- **Semantyka –** zależności między symbolicznymi elementami języka programowania a ich znaczeniem, interpretacją i stosowaniem;
- **Pojedynczy element danych –** element danych zawierający tylko jedną wartość;
- **Krok –** sytuacja, w której jednostka organizacyjna programu zachowuje się względem swoich wejść i wyjść według reguł zdefiniowanych przez akcje związane z tym krokiem;
- **Typ danych strukturalnych –** typ danej agregatu, która została zadeklarowana z zastosowaniem deklaracji STRUCT lub FUNCTION_BLOCK;
- **Indeksowanie –** mechanizm określający element tablicy za pomocą powołania się tablicy i jednego lub więcej wyrażeń, które od momentu określenia ich wartości wskazują pozycję tego elementu;
- **Reprezentacja symboliczna –** stosowanie identyfikatorów do nazywania zmiennych;
- **Zadanie –** element sterowania wykonaniem umożliwiający okresowe lub wyzwalane wykonanie grupy połączonych jednostek organizacyjnych programu;

- **Literał czasowy –** literał reprezentujący dane typu TIME, DATE, TIME_OF_DAY lub DATE_AND_TIME;
- **Przejście –** warunki przechodzenia sterowania od jednego lub więcej kroków poprzedzających do jednego lub więcej kroków następnych, wzdłuż ścieżki programu;
- **Liczba całkowita bez znaku –** literał całkowity nie rozpoczynający się ani od znaku plus (+) ani od znaku minus (-);
- **Łączone OR –** konstrukcja pozwalająca osiągnąć boolowską funkcję OR w języku schematów drabinkowych, poprzez połączenie prawych końcówek połączeń poziomych z połączeniami pionowymi.

### 3.2. Elementy języków (metod) programowania sterowników PLC

Trzecia część normy IEC 1131 wyróżnia następujące elementy języków programowania sterowników PLC:

- Typy danych (ang. *Data Types*);
- Jednostki organizacyjne oprogramowania **POU** (ang. ***P**rogram **O**rganization **U**nits*);
- Elementy schematu (algorytmu) funkcji sekwencyjnej **SFC** (ang. ***S**equential **F**unction **C**hart*);
- Elementy konfiguracji (ang. *Configuration elements*).

***Typy danych*** służą do określenia struktury danych w sterowniku PLC, zarówno danych stałych jak i zmiennych, a w szczególności zakresu wartości, jakie mogą one przyjmować oraz obszaru pamięci, który potrzebny jest do ich przechowywania.

***Jednostki organizacyjne oprogramowania POU*** stanowią najmniejsze niezależne jednostki oprogramowania aplikacji użytkownika. Na POU składają się następujące elementy:

- Funkcje (ang. *Functions*);
- Bloki funkcjonalne (ang. *Function blocks*);
- Programy (ang. *Programs*).

***Elementy konfiguracji*** wspomagają instalowanie i uruchamianie programów użytkownika w systemach sterowników PLC. Zaliczono do nich następujące elementy:

- Konfiguracja (ang. *Configuration*) - elementem języka programowania sterownika PLC, który odpowiada systemowi sterowników PLC rozumianemu jako całość, obejmującą wszystkie pozostałe elementy oprogramowania;
- Zasób (ang. *Resource*) - programowy odpowiednik sprzętu, który realizuje funkcje przetwarzania sygnałów, łącznie z funkcjami określonymi przez podłączone czujniki (ang. *sensors*) i elementy wykonawcze (ang. *actuators*) oraz urządzenia operatorskie **MMI** (ang. ***M**an **M**achine **I**nterface*);
- Zadanie (ang. *Task*) - element sterowania wykonaniem programu PLC, umożliwiający okresowe lub wyzwalane wykonanie grupy połączonych jednostek organizacyjnych programu;
- Zmienna globalna (ang. *Global variable*) - zmienna, której zakres jest globalny;
- Ścieżka dostępu (ang. *Access path*) - połączenie symbolicznej nazwy ze zmienną do celów komunikacji otwartej.

### 3.3. Język listy rozkazów IL (STL)

***Język IL*** (ang. ***I****nstruction* ***L****ist*) występujący również pod nazwą metody ***STL*** *(*ang. ***S****tatement* ***L****ist)*, określany jest często jako analog języka niskiego poziomu typu asembler. Tworzenie programu dla sterownika PLC polega na pisaniu wiersz po wierszu (np. w edytorze typu Notatnik) programu, który obejmuje wykorzystanie instrukcji logicznych, arytmetycznych, operacji relacji, operacji na blokach funkcyjnych takich jak: liczniki, czasomierze, komparatory itp.

> **DEFINICJA**
>
> [*Asembler (ang. assembler) - termin związany z programowaniem i tworzeniem kodu maszynowego dla konkretnego mikroprocesora. Symbolizuje tworzenie tego kodu maszynowego na podstawie kodu źródłowego, wykonanego w niskopoziomowym języku programowania, który bazuje na podstawowych operacjach mikroprocesora, gdzie jednemu rozkazowi napisanemu przez programistę odpowiada jeden rozkaz mikroprocesora*]

Należy podkreślić, że traktowanie języka **IL** jako typowego asemblera (np. dla mikroprocesora jednostki **CPU**) jest nie do końca trafione. Wynika to z zasadniczego faktu, iż tworzenie programu sterującego (według powyższej definicji) w asemblerze np. dla rodziny mikroprocesorów o oznaczeniu 8051 firmy INTEL polega na używaniu instrukcji, które przekładają się bezpośrednio na pojedyncze rozkazy dla takiego mikroprocesora, zaś tworzenie programu sterującego w metodzie **STL** dla sterownika PLC, którego jednostka **CPU** zawiera ten sam mikroprocesor 8051 nie przekłada się na pojedyncze rozkazy dla takiego mikroprocesora, tylko interpretacja poszczególnych instrukcji tego języka **IL** zależy od systemu operacyjnego, który został zaimplementowany w tym sterowniku PLC. Nie występuje odpowiedniość pojedynczego rozkazu mikroprocesora **CPU** w stosunku do pojedynczego rozkazu języka **IL** tak, jak to ma miejsce, przypomnijmy, w języku asemblera. Można zasadnie stwierdzić, że w języku asemblera mamy do czynienia z rozkazami dla mikroprocesora, zaś w języku **IL** mamy do czynienia z poleceniami dla systemu operacyjnego sterownika PLC. Jednak w obu przypadkach, tj. w języku asemblera i w języku IL (STL) musi istnieć tzw. listę rozkazów (ang. ***I****nstruction* ***L****ist*).

⇒ **Rozkazy w języku IL**

Według normy IEC 1131-3 listę rozkazów tworzy sekwencja rozkazów. Każdy rozkaz powinien zaczynać się od nowej linii, zawierać operator (tzw. mnemonik) z opcjonalnymi modyfikatorami i jeżeli jest to niezbędne dla konkretnej operacji, powinien zawierać jeden lub większą liczbę operandów (tzw. argumentów), rozdzielonych przecinkami. Operandy mogą mieć postać dowolnej reprezentacji danych zdefiniowanych w normie. Dla danych, które reprezentują w programie sterownika PLC dane zewnętrzne, przewidziano tzw. literały, zaś dla danych, które reprezentują pewien sposób identyfikowania obiektów danych, których zawartość może się zmieniać, np. dane związane z wejściami, wyjściami czy pamięcią sterownika PLC, przewidziano zmienne. Przykładową reprezentację danych zewnętrznych - literałów ilustrują tabele 12 do 14.

*Tabela 1: Właściwości literałów w postaci liczbowej*

| Opis | Przykłady |
|---|---|
| Liczby całkowite | *-12 0 123_456 +986* |
| Liczby rzeczywiste | *-12.0 0.0 0.456 3.14159_26* |
| Liczby rzeczywiste z wykładnikami | *-1.34E-12 lub -1.34e-12 1.234E6 lub 1.234e6* |
| Liczby dwójkowe | *2#1111_1111 (255) 2#1110_0000 (240)* |
| Liczby ósemkowe | *8#377 (255) 8#340 (240)* |
| Liczby szesnastkowe | *16#FF lub 16#ff (255) 16#E0 lub 16#e0 (240)* |
| Boolowskie zero i jedynka | *0 1* |
| Boolowskie | *FAŁSZ i PRAWDA FALSE TRUE* |

*Tabela 2: Właściwości literałów w postaci czasów trwania*

| Opis | Przykłady |
|---|---|
| Literały bez podkreśleń z przedrostkiem krótkim | *T#14ms T#14.7s T#14.7m T#14.7h t#14.7d t#5d14h12m18s3.5ms* |
| Literały bez podkreśleń z przedrostkiem długim | *TIME#14ms time#14.7s* |
| Literały z podkreśleniem z przedrostkiem krótkim | *T#14ms T#14.7s T#14.7m T#14.7h t#25h_15m t#5d_14h_12m_18s_3.5ms* |
| Literały z podkreśleniem z przedrostkiem długim | *TIME#25h_15m time#5d_14h_12m_18s_3.5ms* |

*Tabela 3: Właściwości literałów daty i czasu dnia:*

| Opis | Przykłady |
|---|---|
| Literał daty (długi przedrostek) | DATE# |
| Literał daty (krótki przedrostek) | D# |
| Literał czasu dnia (długi przedrostek) | TIME_OF_DAY# |
| Literał czasu dnia (krótki przedrostek) | TOD# |
| Literał daty i czasu dnia (długi przedrostek) | DATE_AND_TIME# |
| Literał daty i czasu dnia (krótki przedrostek) | DT# |

W odniesieniu do deklaracji zmiennych w programie dla sterownika PLC należy stwierdzić, że zmienna może być deklarowana jako przynależąca do jednego z typów elementarnych lub do jednego z typów pochodnych. Elementarne typy danych zgodnie z normą IEC 1131-3 ilustruje tablica 15, zaś pochodne typy danych Tablica 16.

*Tabela 4: Elementarne typy danych zgodnie z normą IEC 1131-3*

| Lp. | Słowo kluczowe | Typ danych | Liczba bitów |
|---|---|---|---|
| **1.** | BOOL | Boolowski | **1** |
| **2.** | SINT | Liczba całkowita krótka | **8** |
| **3.** | INT | Liczba całkowita | **16** |
| **4.** | DINT | Liczba całkowita podwójnej długości | **32** |
| **5.** | LINT | Liczba całkowita długa | **64** |
| **6.** | USINT | Liczba całkowita krótka bez znaku | **8** |
| **7.** | UINT | Liczba całkowita bez znaku | **16** |
| **8.** | UDINT | Liczba całkowita podwójnej długości bez znaku | **32** |
| **9.** | ULINT | Liczba całkowita długa bez znaku | **64** |
| **10.** | REAL | Liczby rzeczywiste | **32** |
| **11.** | LREAL | Liczby rzeczywiste długie | **64** |
| **12.** | TIME | Czas trwania | **UWAGA 1** |
| **13.** | DATE | Data (wyłącznie) | **UWAGA 1** |
| **14.** | TIME_OF_DAY lub TOD | Czas dnia (wyłącznie) | **UWAGA 1** |
| **15.** | DATE_AND_TIME lub DT | Data i czas dnia | **UWAGA 1** |
| **16.** | STRING | Ciąg znaków o zmiennej długości | **UWAGA 1** |
| **17.** | BYTE | Ciąg bitów o długości 8 | **8** |
| **18.** | WORD | Ciąg bitów o długości 16 | **16** |
| **19.** | DWORD | Ciąg bitów o długości 32 | **32** |
| **20.** | LWORD | Ciąg bitów o długości 64 | **64** |
| **UWAGA 1: Długość tego typu elementarnych danych jest zależna od implementacji.** | | | |

*Tabela 5: Pochodne typy danych zgodnie z normą IEC 1131-3*

| Lp. | Właściwość / przykład tekstowy |
|---|---|
| **1.** | Alias, nowa nazwa dla typu elementarnego:<br>**TYPE R: REAL; END_TYPE** |
| **2.** | Typ wyliczeniowy (ang. Enumerated data type):<br>**TYPE ANALOG_SIGNAL_TYPE: (SINGLE_ENDED, DIFFERENTIAL); END_TYPE** |
| **3.** | Typ okrojony (ang. Subrange data type)<br>**TYPE ANALOG_DATA: INT ( -4095...4095); END_TYPE** |
| **4.** | Deklaracja tablicy danych (ang. Array data type)<br>**TYPE ANALOG_16_INPUT: ARRAY [1...16] OF ANALOG_DATA; END_TYPE** |
| **5.** | Deklaracje typu strukturalnego (ang. Structured data type)<br>**TYPE**<br>**ANALOG_CHANNEL_CONFIG:**<br>**STRUCT**<br>**RANGE: ANALOG_RANGE;**<br>**MIN_SCALE: ANALOG_DATA;**<br>**MAX_SCALE: ANALOG_DATA;**<br>**END_STRUCT;**<br>**ANALOG_16_INPUT_CONFIG:**<br>**STRUCT**<br>**SIGNAL_TYPE: ANALOG_SIGNAL_TYPE;**<br>**FILTER_PARAMETER: SINT (0...99);**<br>**CHANNEL: ARRAY [1...16] OF ANALOG_CHANNEL_CONFIG;**<br>**END_STRUCT;**<br>**END_TYPE** |

Rozkaz w języku **IL** może być poprzedzony etykietą, która wtedy pełni rolę identyfikatora, po której jest umieszczany dwukropek „**:**”. Jeśli występuje komentarz, to powinien on być ostatnim elementem wiersza. Norma dopuszcza, aby między rozkazami mogły być wprowadzane puste wiersze. Przykłady rozkazów w języku **IL** ilustruje Tabela 17.

*Tabela 6: Przykłady użycia rozkazów w języku IL (STL)*

| ETYKIETA | OPERATOR | OPERAND | KOMENTARZ |
|---|---|---|---|
| START: | LD | %IX1 | (*Wciśnij przycisk*) |
| | ANDN | %MX5 | (*Nie zatrzymane*) |
| | ST | %QX2 | (*Włączone*) |

⇒ **Operatory, modyfikatory i operandy w języku IL**

Standardowe operatory z dopuszczalnymi ich modyfikacjami powinny mieć postać taką, jaką wyszczególnia Tabela 18. Jeżeli nie podano inaczej, to semantykę operatora określa poniższe wyrażenie:

***wynik:= wynik OP operand***

Powyższy zapis oznacza, że wyznaczana wartość wyrażenia zastępowana jest jego bieżącą wartością, przetworzoną przez operator z uwzględnieniem operandu. Przykładowo rozkaz AND %IX1 wygląda jak poniżej:

***wynik:= wynik AND %IX1***.

*Tabela 7: Operatory języka IL (STL)*

| Lp. | Operator | Modyfikatory | Operand | Semantyka |
|---|---|---|---|---|
| **1.** | LD | N | UWAGA 2 | Ustaw bieżącą wartość wyniku równą operandowi |
| **2.** | ST | N | UWAGA 2 | Zapamiętaj bieżący wynik w miejscu operandu |
| **3.** | S | UWAGA 3 | BOOL | Ustaw operand boolowski na „1” |
| | R | UWAGA 3 | BOOL | Ustaw operand boolowski na „0” (jeśli był na „1”) |
| **4.** | AND | N, ( | BOOL | Boolowskie AND |
| **5.** | & | N, ( | BOOL | Boolowskie AND |
| **6.** | OR | N, ( | BOOL | Boolowskie OR |
| **7.** | XOR | N, ( | BOOL | Boolowskie ExOR |
| **8.** | ADD | ( | UWAGA 2 | Dodawanie |
| **9.** | SUB | ( | UWAGA 2 | Odejmowanie |
| **10.** | MUL | ( | UWAGA 2 | Mnożenie |
| **11.** | DIV | ( | UWAGA 2 | Dzielenie |
| **12.** | GT | ( | UWAGA 2 | Porównanie: > |
| **13.** | GE | ( | UWAGA 2 | Porównanie: >= |
| **14.** | EQ | ( | UWAGA 2 | Porównanie: = |
| **15.** | NE | ( | UWAGA 2 | Porównanie: <> |
| **16.** | LE | ( | UWAGA 2 | Porównanie: <= |
| **17.** | LT | ( | UWAGA 2 | Porównanie: < |
| **18.** | JMP | C, N | LABEL | Skocz do etykiety |
| **19.** | CAL | C, N | NAME | Wywołaj blok funkcji (UWAGA 4) |
| **20.** | RET | C, N | | Powróć z wywołanej funkcji lub bloku funkcji |
| | ) | | | Wykonaj wstrzymaną operację |

**UWAGI:**

UWAGA 2: Operatory powinny być albo przeciążone albo z nadanym typem. Bieżący wynik i operand powinny być tego samego typu.

UWAGA 3: Operacje te są wykonywane wtedy i tylko wtedy, gdy bieżąca wartość wyniku jest równa boolowskiej „1”.

UWAGA4: Po nazwie bloku funkcji występuje umieszczona w nawiasach zwykłych lista argumentów.

⇒ **Funkcje i bloki funkcji języka IL (STL)**

Funkcje można wywołać poprzez umieszczenie nazwy funkcji w polu operatora. Wynik bieżący może być wykorzystywany jako pierwszy argument funkcji. Argumenty dodatkowe, jeśli są wymagane, powinny być podane w polu operandu. Wartość określona przez funkcję, w wyniku pomyślnego wykonania rozkazu RET lub po osiągnięciu fizycznego końca funkcji, może stać się wynikiem bieżącym.

Bloki funkcji mogą być wywołane warunkowo lub bezwarunkowo za pomocą operatora CAL (wywołanie). Wywołanie to może mieć jedną z trzech postaci, które podano w Tabeli 19, zaś standardowe operatory wejściowe bloków funkcji języka IL (STL) pokazuje Tabela 20.

*Tabela 8: Właściwości wywołania bloków funkcji dla języka IL (STL)*

| Lp. | Opis wywołania bloków funkcji/przykład |
|---|---|
| **1.** | CAL z listą wejść:<br>**CAL C10 (CU:= %IX10, PV:= 15)** |
| **2.** | CAL z załadowaniem/zapamiętaniem wejść:<br>**LD 15**<br>**ST C10.PV**<br>**LD %IX10**<br>**ST C10.CU**<br>**CAL C10** |
| **3.** | Użycie operatorów wejściowych:<br>**LD 15**<br>**PV C10**<br>**LD %IX10**<br>**CU C10** |
| UWAGA:<br>W powyższych przykładach zakłada się istnienie takich deklaracji jak VAR C10:= CTU; END_VAR. | |

*Tabela 9: Standardowe operatory wejściowe bloków funkcji języka IL (STL)*

| Lp. | Operatory | Typ FB |
|---|---|---|
| **1.** | **S1, R** | **SR** |
| **2.** | **S, R1** | **RS** |
| **3.** | **CLK** | **R_TRIG** |
| **4.** | **CLK** | **F_TRIG** |
| **5.** | **CU, R, PV** | **CTU** |
| **6.** | **CD, LD, PV** | **CTD** |
| **7.** | **CU, CD, R, LD, PV** | **CTUD** |
| **8.** | **IN, PT** | **TP** |
| **9.** | **IN, PT** | **TON** |
| **10.** | **IN, PT** | **TOF** |

⇒ **Przykład programu dla sterownika PLC w języku IL (STL)**

**Network 1**

| | |
|---|---|
| LD | **I0.0** |
| ON | **I0.1** |
| AN | **I0.7** |
| = | **M0.0** |

**Network 2**

| | |
|---|---|
| LD | **M0.0** |
| AN | **I0.3** |
| = | **Q0.0** |

**Network 3**

| | |
|---|---|
| LD | **I0.1** |
| OR | **M0.1** |
| = | **M0.1** |
| = | **Q0.1** |

**Network 4**

| | |
|---|---|
| LD | **I0.7** |
| A | **I0.5** |
| A | **M0.5** |
| A | **T35** |
| OR | **M0.7** |
| = | **Q0.4** |

### 3.4. Język ST (Język tekstu strukturalnego)

Zgodnie z normą IEC 1131-3 Język **ST** (ang. **S**tructural **T**ext) jest odpowiednikiem języka algorytmicznego wysokiego poziomu, zawierający struktury programowe i polecenia podobne do występujących w językach typu PASCAL lub C.

> **DEFINICJA**
>
> [*Język wysokiego poziomu – typ języka programowania, którego składnia i słowa kluczowe mają maksymalnie ułatwić rozumienie kodu programu, jednocześnie dystansując się od wymagań sprzętowych*]

Składnikiem języka **ST** jest wyrażenie. Jest ono konstrukcją języka, która po wyznaczeniu jej wartości wytwarza wartość zgodną z jednym z typów danych, określonych w p.3.3.3. Wyrażenia są tworzone z operatorów oraz operandów. Operatorem może być literał, którego definicje podano w p.3.3.3 jak również zmienna może być zmienną zgodnie z p.3.3.3.

⇒ **Operatory języka ST**

Wyznaczanie wartości wyrażenia obejmuje zastosowanie operatorów do operandów w kolejności wynikającej z określonego w normie pierwszeństwa operatorów. Najpierw stosowany jest operator mający najwyższe pierwszeństwo (priorytet) w wyrażeniu, potem stosowany jest operator o kolejnym niższym pierwszeństwie, itd., aż do zakończenia procesu wyznaczania wartości. Operatory języka **ST** zawarto w Tabeli 21.

*Tabela 10: Operatory języka ST*

| Lp. | Operacja | Symbol | Priorytet |
|---|---|---|---|
| **1.** | Wyrażenie nawiasowe | **(Wyrażenie)** | **NAJWYŻSZY** |
| **2.** | Obliczanie wartości funkcji<br>Przykłady: | **Identyfikator**<br>**LN(A), MAX(X, Y), itd.** | |
| **3.** | Potęgowanie | ****** | |
| **4.**<br>**5.** | Negacja<br>Uzupełnienie | **-**<br>**NOT** | |
| **6.**<br>**7.**<br>**8.** | Mnożenie<br>Dzielenie<br>Modulo | *****<br>**/**<br>**MOD** | |
| **9.**<br>**10.** | Dodawanie<br>Odejmowanie | **+**<br>**-** | |
| **11.** | Porównanie | **<, >, <=, >=** | |
| **12.**<br>**13.** | Równość<br>Nierówność | **=**<br>**<>** | |
| **14.**<br>**15.** | Boolowskie AND<br>Boolowskie AND | **&**<br>**AND** | |
| **16.** | Boolowskie wykluczające OR (ExOR) | **XOR** | |
| **17.** | Boolowskie OR | **OR** | **NAJNIŻSZY** |

Norma IEC 1131-3 stanowi, że operatory o jednakowym pierwszeństwie mogą być stosowane w kolejności ich zapisu w wyrażeniu, począwszy od strony lewej do prawej. Na przykład, gdy A=1, B=2, C=3 i D=4, czy są typu INT, to wyrażenie ***A + B – C * ABS(D)*** będzie miało wartość „**-9**”, natomiast wyrażenie przy tych samych danych ***(A + B – C) * ABS(D)*** będzie miało wartość „**0**”.

Jeśli operator ma dwa operandy, najpierw określana jest wartość operandu w skrajnym lewym położeniu. Na przykład w wyrażeniu ***SIN(A) * COS(B)*** najpierw wyznaczana będzie wartość wyrażenia SIN(A), następnie COS(B), po czym dopiero wartość iloczynu.

Wartości wyrażeń boolowskich mogą być określone tylko w zakresie niezbędnym do wyznaczania wartości wynikowej. Na przykład, gdy ***A<=B***, to wyznaczając tylko wartość wyrażenia ***(A>B)*** można stwierdzić, że wartość wyrażenia ***(A>B) & (C<D)*** jest równa boolowskiej zero.

⇒ **Instrukcje języka ST**

Instrukcje języka **ST** zgodne z normą IEC 1131-3 podano w Tabeli 22. Norma stanowi, że instrukcje te powinny być w programie dla sterownika PLC zakończone średnikami „ **;** ”.

*Tabela 11: Instrukcje języka ST*

| Lp. | Typ instrukcji/Odniesienie | Przykłady użycia |
|---|---|---|
| **1.** | Przypisanie | **A := B; CV := CV+1; Y := SIN(X); D := INT_TO_REAL(C)** |
| **2.** | Wywołanie bloku funkcjonalnego (funkcji) oraz użycie wyjścia FB | **CMD_TMR(IN:=%IX5, PT:= T#300ms);**<br>**A:=CMD_TMR.Q;** |
| **3.** | **RETURN** | **RETURN;** |
| **4.** | **IF** | **D:= B*B - 4*A*C; (* oblicz wyróżnik *)**<br>**IF D < 0.0 THEN NROOTS:= 0; (* brak pierwiastków *)**<br>**ELSIF D = 0.0 THEN (* jeden pierwiastek *)**<br>**NROOTS := 1;**<br>**X1 := – B / (2.0*A);**<br>**ELSE (* dwa pierwiastki *)**<br>**NROOTS := 2;**<br>**X1 := (–B + SQRT(D))/(2.0*A);**<br>**X2 := (–B – SQRT(D))/(2.0*A);**<br>**END_IF;** |
| **5.** | **CASE** | **ERROR:=0; (* zmienna boolowska *)**<br>**XW:=BCD_TO_INT(Y); (* wyznacz wartość wybieraka *)**<br>**CASE XW OF (* instrukcja wyboru *)**<br>**1,4 : DISPLAY := TEKST1;**<br>**2 : DISPLAY := TEKST2; (* blok instrukcji )**<br>**Y := SIN(Z); (* kolejne instrukcje, gdy XW = 2 *)**<br>**3,5..10: DISPLAY := STATUS (XW – 3);**<br>**ELSE DISPLAY := ’’; (* XW poza zakresem 1..10 *)**<br>**ERROR := 1;**<br>**END_CASE; (* koniec instrukcji wyboru *)** |
| **6.** | **FOR** | **J := 101;**<br>**FOR I := 1 TO 100 BY 2 DO**<br>**IF WORDS(I) = ’KEY’ THEN**<br>**J := I;**<br>**EXIT;**<br>**END_IF;**<br>**END_FOR;** |
| **7** | **WHILE** | **J := 1;**<br>**WHILE J <= 100 AND WORDS(J) <> ’KEY’ DO**<br>**J := J+2;**<br>**END_WHILE;** |
| **8.** | **REPEAT** | **J := –1;**<br>**REPEAT**<br>**J := J+2;**<br>**UNTIL J = 101 OR WORDS(J) = ’KEY’**<br>**END_REPEAT;** |
| **9.** | **EXIT** | **EXIT;** |
| **10** | Instrukcja pusta | **;** |

**UWAGA:**
Jeśli jest realizowana instrukcja EXIT, to powinna być ona zaimplementowana we wszystkich instrukcjach FOR, WHILE, REPEAT, które są realizowane w danej implementacji.

⇒ **Opis instrukcji języka ST**

**Ad1:** Instrukcja przypisania zastępuje bieżącą wartość pojedynczej lub wieloelementowej zmiennej wynikiem procesu określania wartości wyrażenia. Instrukcja przypisania powinna zawierać odniesienie do zmiennej, która znajduje się z lewej strony po której następuje operator przypisania „ **:=** „, a następnie wyrażenie, którego wartość będzie wyznaczana. Na przykład instrukcja ***A := B***; będzie wykorzystana do zastąpienia pojedynczej danej zmiennej **A** bieżącą wartością **B**, w przypadku gdy obie te wartości są typu INT.

**Ad2:** Instrukcje sterujące funkcji i bloku funkcji składają się z mechanizmów wywoływania bloków funkcji FB oraz zwracania sterowania jednostce wywołującej przed fizycznym końcem funkcji lub bloku funkcji. Takie wywołanie może stanowić część wyznaczania wartości wyrażenia.

Bloki funkcji FB mogą być wywołane za pomocą instrukcji zawierającej nazwę bloku funkcji, za którą umieszczona jest w nawiasie lista wartości przypisanych nazwanym parametrom wejściowym, jak podaje Tablica 22. Kolejność, w jakiej parametry wejściowe są wyspecyfikowane w wywołaniu bloku funkcji nie powinna być istotna. Nie jest wymagane, aby wszystkim parametrom wejściowym były przypisywane wartości przy każdym wywołaniu bloku funkcji. Jeśli w wywołaniu bloku funkcji określonemu parametrowi nie jest przypisana wartość, to powinna być stosowana wartość poprzednia lub początkowa, jeżeli nie było żadnego wcześniejszego przypisania.

**Ad3:** Instrukcja ***RETURN*** stwarza możliwość wcześniejszego wyjścia z funkcji lub bloku funkcji, np. w rezultacie określenia wartości instrukcji IF.

**Ad 4:** Instrukcja ***IF*** jest jedną z instrukcji wyboru, tzw. warunkowe rozgałęzienie programu. Instrukcja ta określa, że grupa instrukcji będzie wykonywana tylko wówczas gdy wartość związanego z nią wyrażenia boolowskiego (tzw. warunku) będzie wynosiła „**1**”. Jeżeli wynikiem sprawdzenia tego warunku będzie „**0**”, to nie będzie wykonywana żadna instrukcja lub będzie wykonana grupa instrukcji występująca po słowie kluczowym ELSE lub po słowie kluczowym ELSIF, jeżeli związany warunek boolowski będzie równy „**1**”.

**Ad 5:** Instrukcja ***CASE*** jest drugą z tzw. instrukcji wyboru. Instrukcja ta zawiera wyrażenie, które może być sprowadzone do zmiennej typu INT („selektor”) oraz listy grup instrukcji, przy czym każda grupa jest oznakowana jedną lub wieloma liczbami całkowitymi lub zakresami wartości całkowitych. W ten sposób określa się, że będzie wykonana pierwsza grupa instrukcji, w której zakresie znajduje się wyliczona wartość. Jeżeli wartość nie występuje w żadnym z zakresów to wykonywana będzie sekwencja instrukcji znajdująca się za słowem kluczowym ELSE, w przypadku, gdy występuje ono w instrukcji ***CASE***. W innych przypadkach nie będzie wykonywana żadna z sekwencji instrukcji.

**Ad6÷8:** Instrukcje ***FOR***, ***WHILE*** oraz ***REPEAT*** należą do grupy tzw. instrukcji iteracji, czyli czynności powtarzania tej samej operacji w pętli przez z góry określoną liczbę razy lub aż do spełnienia określonego warunku

(boolowskiego). W instrukcjach tych określono, że grupa związanych z nimi instrukcji powinna być wykonywana repetycyjnie. Instrukcja ***FOR*** jest wykorzystywana wówczas, gdy liczba iteracji może być określona wcześniej, natomiast w innych przypadkach są stosowane instrukcje ***WHILE*** lub ***REPEAT***.

**Ad9:** Instrukcja ***EXIT*** może być wykorzystana do zakończenia iteracji przed spełnieniem warunku zakończenia iteracji. Jeśli instrukcja ***EXIT*** znajduje się wewnątrz zagnieżdżonej konstrukcji iteracyjnej, wyjście powinno nastąpić z najbardziej wewnętrznej pętli, w której ta instrukcja jest umieszczona, zaś sterowanie powinno być przekazane do instrukcji znajdującej się bezpośrednio za znakiem końca pierwszej pętli (END_FOR, END_WHILE lub END_REPEAT), występującym po instrukcji ***EXIT***. Przykład użycia instrukcji ***EXIT*** pokazuje poniższy fragment programu w języku **ST**:

```
SUM := 0;
FOR I := 1 TO 3 DO
   FOR J := 1 TO 2 DO
        IF FLAG THEN EXIT; END_IF
        SUM := SUM + J;
   END_FOR;
   SUM := SUM + I;
END_FOR;
```

W wykonaniu instrukcji przedstawionych w powyższym fragmencie programu, który został utworzony w języku ST, wartość zmiennej SUM będzie wynosiła 15 wówczas, gdy wartość zmiennej boolowskiej FLAG wyniesie „**0**”, zaś wynosić 6, gdy wartość zmiennej boolowskiej FLAG wyniesie „**1**”.

⇒ **Przykład programu dla sterownika PLC w języku ST**

```
VAR_INPUT
We : BOOL;
Y : INT;
Yzad : INT;
Tz : TIME := t#100ms;
Tw : TIME := t#100ms;
END_VAR
VAR_OUTPUT
Q : BOOL;
END_VAR
VAR
Timer_ON : TON;
Timer_OFF : TON;
FlipFlop : SR;
END_VAR
IF We THEN
   IF (Y < Yzad) THEN
   Timer_ON(IN: = We, PT := Tz);
END_IF;
```

### 3.5. Język (schemat drabinkowy) LD (LAD)

Semantyka pierwszego z języków graficznych - **LD** (**LAD**) (ang. ***L**adder **D**iagram*), narzucona w trzeciej części normy IEC 1131 spowodowała, że jest on swoją postacią graficzną zbliżony do konstrukcji schematów elektrycznych, z wykorzystaniem m.in. obwodów przekaźnikowych z zastosowaniem ich zestyków. Uczyniono tak, aby można było stosunkowo łatwo zamienić to powszechnie stosowane sterowanie elektryczne na program dla sterownika PLC. Rozszerzeniem języka **LD** w stosunku do swojego „protoplasty" była możliwość używania różnych funkcji arytmetycznych, logicznych, porównań, relacji, itp. Wizualnie konstrukcja programu sterującego dla sterownika PLC w metodzie **LD** odróżnia się od schematu elektrycznego (stykowego) tym, iż program PLC tworzony jest na arkuszu (czytaj: ekranie programatora) niejako od lewej strony do prawej (tzn. „zestyki" programu w postaci krótkich dwóch pionowych linii występują po lewej stronie arkusza, zaś umowne „cewki" po prawej), zaś tworzenie schematu elektrycznego polega na rysowaniu obwodów niejako z góry na dół (zestyki występuję w górnej części arkusza, zaś cewki dolnej).

Norma IEC 1131-3 w zakresie języków (metod) graficznych, do których oprócz języka LD opracowano jeszcze dwa języki: FBD oraz SFC (będzie o nich mowa dalej), nakreśliła tzw. elementy graficzne wspólne, które mają zastosowanie w tworzeniu programu dla sterownika PLC w tych trzech językach programowania.

⇒ **Elementy wspólne dla języków IL, FBD i SFC**

Norma zdefiniowała sieć programową jako maksymalny zbiór połączonych ze sobą elementów graficznych. Przepływ wielkości pojęciowych przez jedną lub większą liczbę sieci programowych, które reprezentują system sterowania w językach **IL**, **FBD** oraz **SFC** może być rozpatrywany jako odpowiednio:

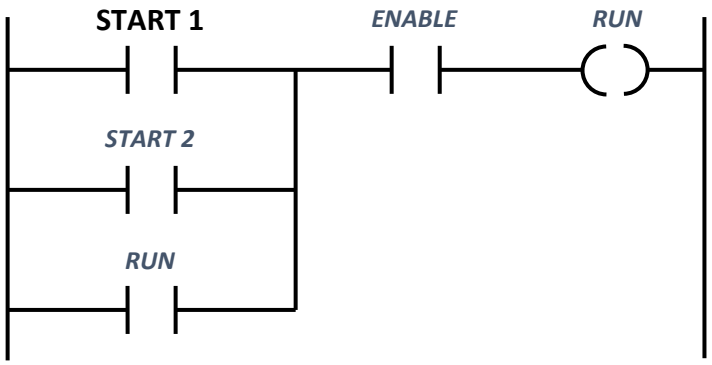
- „Przepływ mocy", analogicznie jak przepływ energii elektrycznej w systemach z przekaźnikami elektromechanicznymi, stosowany w schematach stykowych;
- „Przepływ sygnału", analogicznie jak przepływ sygnałów między elementami systemu przetwarzania sygnału, stosowany w schematach sterowania cyfrowego;
- „Przepływ działań", analogicznie jak przepływ sterowania między elementami organizacji lub między krokami sekwensera elektromechanicznego, stosowany w schematach funkcji sekwencyjnych.

Odpowiednie wielkości pojęciowe, czyli rodzaj abstrakcji, która pozwala na analizę programu w językach graficznych, mogą „przepływać" wzdłuż linii między elementami sieci programowych, zgodnie z następującymi regułami:

1. Przepływ mocy w języku LD powinien odbywać się od strony lewej do prawej;
2. Przepływ sygnału w języku FBD powinien odbywać się od strony wyjścia (z prawej strony) bloku funkcji do wejścia (z lewej strony) funkcji lub bloku (bloków) funkcji połączonych ze sobą w taki sposób;
3. Przepływ działań między elementami SFC powinien odbywać się od dolnej części kroku, poprzez odpowiednie przejście do wierzchołka następnego odpowiedniego kroku (kroków).

Norma IEC 1131-3 określa, że istnienie tzw. ścieżki sprzężenia zwrotnego w sieci programowej stwierdza się wówczas, gdy wyjście funkcji lub bloku funkcjonalnego jest wykorzystywane jako wejście funkcji lub bloku

funkcjonalnego, które wystąpiły wcześniej w sieci programowej, przy czym zmienna skojarzona z tą ścieżką jest nazywana zmienną sprzężenia zwrotnego. Pojęcie ścieżki sprzężenia zwrotnego w języku LD ilustruje rys. 40.




*Rysunek 40: Ścieżka sprzężenia zwrotnego w języku LD*

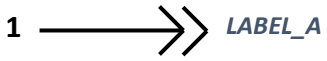
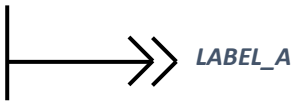
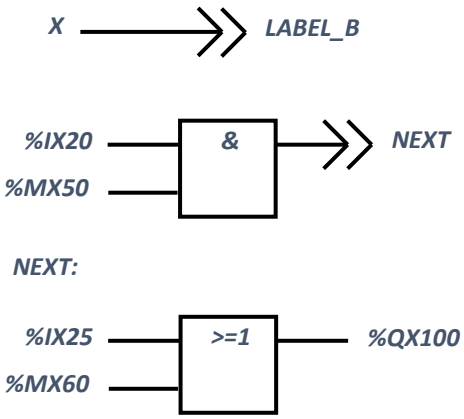
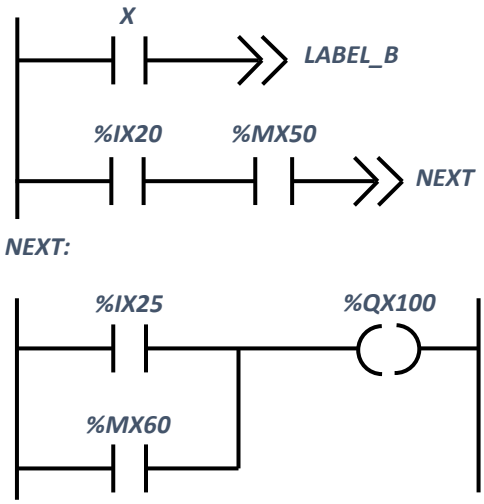
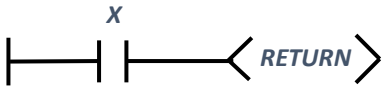
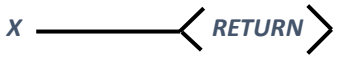
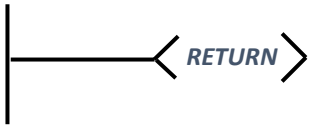
Wyjście oznaczone na rysunku 40 jako „**RUN**" (po prawej stronie) jest wykorzystane jako jedno z trzech wejść (po lewej stronie) łącznie z innymi dwoma, czyli „START1" oraz „START2".

Ścieżki sprzężenia zwrotnego mogą być wykorzystywane w językach graficznych według następujących zasad:

1. Pętle jawne mogą występować tylko w języku FBD;
2. Użytkownik powinien mieć możliwość określania działania elementów w pętli jawnej;
3. Zmienne sprzężenia zwrotnego powinny być odpowiednio inicjowane;
4. Z chwilą, gdy zostanie wyznaczona wartość zmiennej sprzężenia zwrotnego elementu stanowiącego zmienną wyjściową, ta nowa wartość zmiennej sprzężenia zwrotnego powinna być wykorzystywana aż do następnego wyznaczenia wartości elementu.

Norma IEC 1131-3 nakreśliła, że tzw. elementy sterowania wykonaniem utworzonego w językach LD oraz FBD programu powinny być przedstawiane w postaci odpowiednich elementów graficznych. Elementy te ilustruje Tabela 23. Widać w niej, że dla warunku skoku linia sygnału powinna rozpoczynać się w zmiennej boolowskiej, wyjściu boolowskim lub bloku funkcji, lub na linii przepływu mocy języka **LD**. Przekazanie zaś sterowania programem do wyznaczonej etykiety sieci powinno mieć miejsce wówczas, gdy wartość boolowska linii sygnału wynosi „1", zatem skok bezwarunkowy stanowi szczególny przypadek skoku warunkowego.

*Tabela 12: Symbole graficzne elementów sterowania wykonaniem programu PLC*

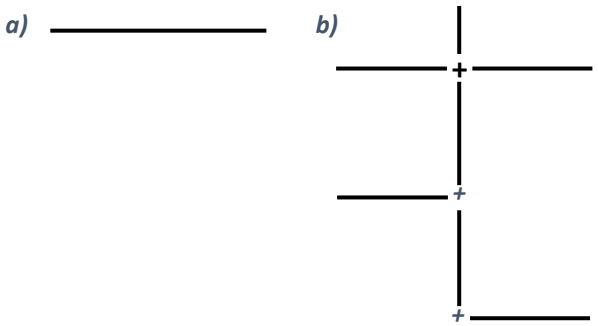
| Lp. | Symbol/Przykład | Wyjaśnienie |
|---|---|---|
| 1. |  | <u>Skok bezwarunkowy:</u><br>Język **FBD** |
| 2. |  | Język **LD** |
| 3. |  | <u>Skok warunkowy:</u><br>Język **FBD**<br><br>Przykład:<br>Warunek skoku<br><br>Cel skoku |
| 4. |  | <u>Skok warunkowy:</u><br>Język **LD**<br><br>Przykład:<br>Warunek skoku<br><br>Cel skoku |
| 5. |  | Powrót warunkowy:<br>Język **LD** |
| 6. |  | Język **FBD** |
| 7. | **END_FUNCTION**<br>**END_FUNCTION_BLOCK** | <u>Powrót bezwarunkowy:</u><br>z FUNCTION |
| 8. |  | z FUNCTION_BLOCK<br>(język **LD**) |

⇒ **Język LD (LAD)**

Norma IEC 1131-3 narzuciła, aby program dla sterownika PLC, który został utworzony w języku LD umożliwiał temu urządzeniu wykonywanie testów oraz modyfikowanie danych za pomocą standardowych symboli graficznych. Symbole, o których mowa rozmieszczane są w tzw. sieciach programowych w sposób podobny do „szczebla" (drabiny) przekaźnikowego schematu drabinkowego (schematu stykowego). Sieci programowe języka LD są ograniczone z lewej i prawej strony przez „szyny zasilania", które są umownym początkiem oraz końcem tej sieci – patrz p.4, pozycja 2 w Tabeli 23. (Umownym, ponieważ dla urządzenia typu sterownik PLC to, co programista tworzy na ekranie programatora w środowisku narzędziowym z językiem LD jest zamieniane na kod cyfrowy dla mikroprocesora tego urządzenia. Szyny zasilające w języku LD mają tylko upodobnić ten język do sposobu tworzenia schematu stykowego). Norma IEC 1131-3 oprócz szyn zasilających wprowadziła jeszcze szereg pojęć, związanych z językiem LD. Są one następujące:

1. Elementy łączące i stany;
2. Styki;
3. Cewki;
4. Funkcje i bloki funkcji;
5. Kolejność wyznaczania wartości wyjściowych w sieciach programowych.

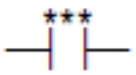
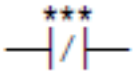
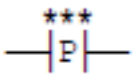
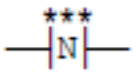
**Ad 1:** Norma stanowi, że elementy łączące w sieciach programowych w języku **LD** mogą przebiegać poziomo lub pionowo. Element łączący powinien znajdować się w stanie „**ON**" lub „**OFF**", zależnie od odpowiadających im wartości boolowskich odpowiednio „**1**" lub „**0**". Określenie „stan łącza" powinno być traktowane jako synonim „przypływu mocy". Zakłada się, że lewa szyna przez cały czas znajduje się w stanie ON, przy czym nie definiuje się stanu prawej szyny. (Często w środowiskach narzędziowych dla różnych sterowników PLC prawa szyna nie występuje). Rysunek 41 ilustruje elementy łączące używane do tworzenia programu w języku **LD**.




*Rysunek 41: Elementy łączące w języku LD:*
*a) poziomy; b) pionowy z poziomym*

**Ad2:** Styk jest elementem przekazującym stan do poziomego elementu łączącego, który znajduje się z jego prawej strony co jest równoważne funkcji boolowskiej AND stanu poziomego elementu łączącego z jego lewej strony i odpowiedniej funkcji skojarzonego boolowskiego wejścia, wyjścia lub zmiennej w pamięci (tzw. Marker). Styk nie może zmieniać wartości sprzężonej zmiennej boolowskiej. Standardowe symbole styków, które używane są w języku LD do tworzenia programu dla sterownika PLC ilustruje Tabela 24.

*Tabela 13: Standardowe symbole styków używane w języku LD*

| STYK | SYMBOL | OPIS |
|---|---|---|
| Styki statyczne |  | *Styk zwierny* (*normalnie otwarty*, ang. *normally open contact*)<br>Stan połączenia z lewej strony styku jest przenoszony na prawą stronę, jeżeli skojarzona zmienna boolowska ma wartość *1*. W przeciwnym razie prawe połączenie jest w stanie *OFF*. |
| |  | *Styk rozwierny* (*normalnie zamknięty*, ang. *normally closed contact*)<br>Stan połączenia z lewej strony styku jest przenoszony na prawą stronę, jeżeli skojarzona zmienna boolowska ma wartość *0*. W przeciwnym razie prawe połączenie jest w stanie *OFF*. |
| Styki impulsowe (wrażliwe na zbocze) |  | *Styk wrażliwy na zbocze narastające*<br>(ang. *Positive transition-sensing contact*)<br>Połączenie z prawej strony styku jest w stanie *ON* w czasie jednego wykonania, jeśli połączenie z lewej strony jest w stanie *ON*, a skojarzona zmienna boolowska zmieniła wartość z *0* na *1*. Poza tym stan połączenia z prawej strony jest *OFF*. |
| |  | *Styk wrażliwy na zbocze opadające*<br>(ang. *Negative transition-sensing contact*)<br>Połączenie z prawej strony styku jest w stanie *ON* w czasie jednego wykonania, jeśli połączenie z lewej strony jest w stanie *ON*, a skojarzona zmienna boolowska zmieniła wartość z *1* na *0*. Poza tym stan połączenia z prawej strony jest *OFF*. |

**Ad 3:** Cewka w języku LD kopiuje bez zmian stan elementu łączącego, który znajduje się z jej lewej strony na stan elementu łączącego, który znajduje się z prawej strony i zapamiętuje odpowiednią funkcję stanu lub przejścia, znajdującego się z lewej strony elementu łączącego na sprzężoną zmienna boolowską. Standardowe symbole cewek podane są w Tabeli 25.

**Ad 4:** Według normy reprezentacja funkcji i bloków funkcji w języku **LD** powinna być zgodna z podaną wcześniej definicją z następującymi wyjątkami:

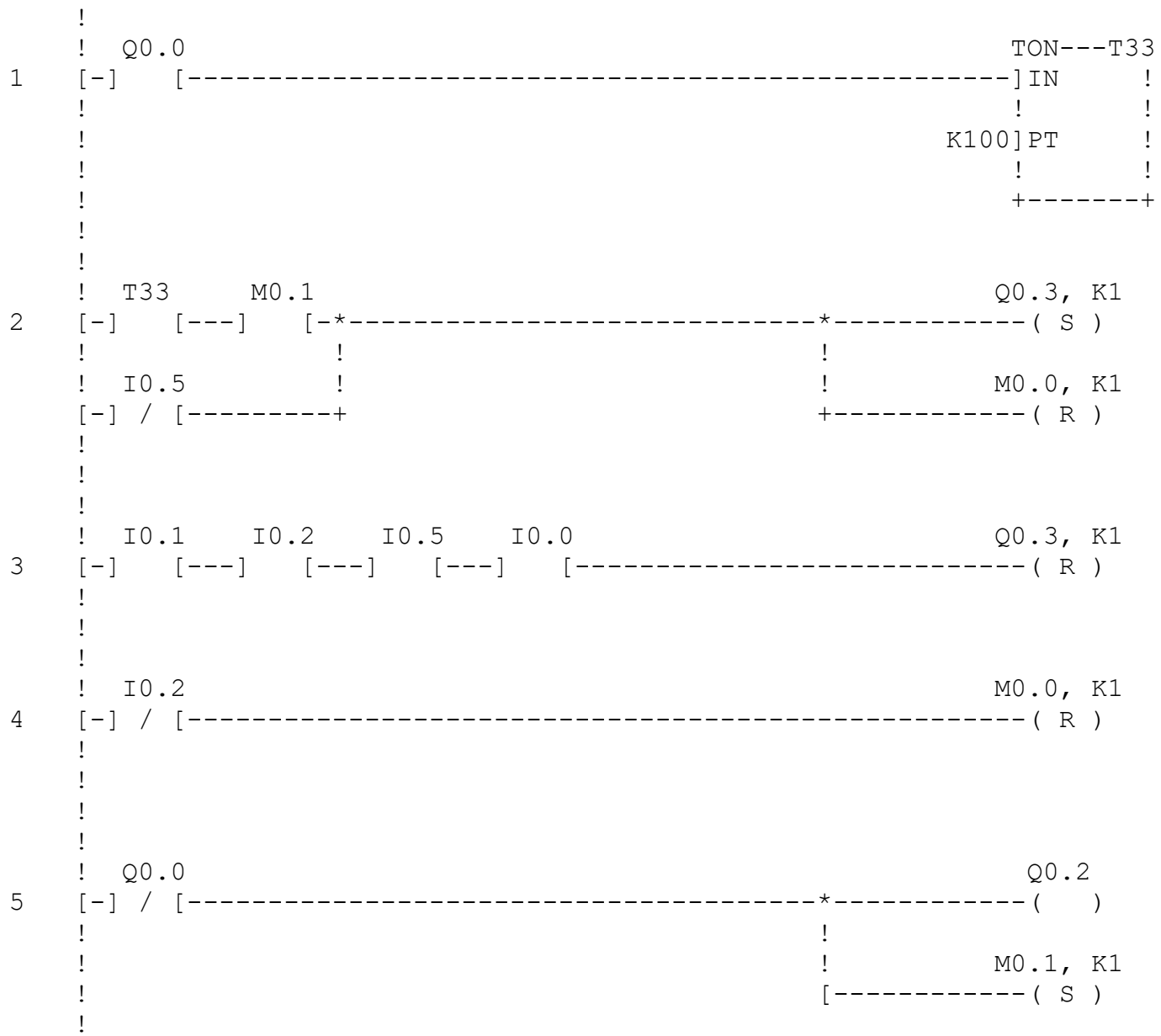
1. Bieżące powiązania parametrów opcjonalnie mogą być przedstawione poprzez pisanie odpowiedniej danej lub zmiennej na zewnątrz bloku, w sąsiedztwie nazwy parametru formalnego wpisanego wewnątrz;
2. W celu umożliwienia przepływu mocy przez blok, w każdym bloku powinno być wskazane co najmniej jedno wejście boolowskie i jedno wyjście boolowskie.

*Tabela 14: Standardowe symbole cewek w języku LD*

| Cewki | Symbol | Opis |
|---|---|---|
| Cewki zwykłe | *** —( )— | *Cewka* (ang. *coil*)<br>Stan połączenia z lewej strony cewki jest przenoszony na prawą stronę i zapamiętywany w skojarzonej zmiennej boolowskiej. |
| | *** —(/)— | *Cewka negująca* (ang. *negated coil*)<br>Stan połączenia z lewej strony cewki jest przenoszony na prawą stronę, a jego odwrotność jest zapamiętywana w skojarzonej zmiennej boolowskiej. |
| Cewki zatrzaskiwane | *** —(S)— | *Cewka ustawiająca* (ang. *Set coil, Latch coil*)<br>Skojarzona zmienna przyjmuje wartość *1*, jeżeli połączenie z lewej strony jest w stanie *ON*. Wartość ta pozostanie niezmieniona, aż do chwili wyzerowania przez cewkę kasującą (R). |
| | *** —(R)— | *Cewka kasująca* (ang. *Reset coil, Unlatch coil*)<br>Skojarzona zmienna przyjmuje wartość *0*, jeżeli połączenie z lewej strony jest w stanie *ON*. Wartość ta pozostanie niezmieniona, aż do chwili ustawienia przez cewkę ustawiającą (S). |
| Cewki podtrzymywane, cewki z pamięcią | *** —(M)— | *Cewka z zapamiętaniem stanu* (ang. *Retentive coil, Memory coil*) |
| | *** —(SM)— | *Cewka ustawiająca z zapamiętaniem stanu*<br>(ang. *Set retentive coil*) |
| | *** —(RM)— | *Cewka kasująca z zapamiętaniem stanu*<br>(ang. *Reset retentive coil*) |
| Cewki impulsowe (wrażliwe na zbocze) | *** —(P)— | *Cewka wrażliwa na zbocze narastające*<br>(ang. *Positive transition-sensing coil*)<br>Skojarzona zmienna przyjmuje wartość *1* na czas jednego wykonania, jeśli połączenie z lewej strony zmieniło stan z *OFF* na *ON*. Stan połączenia z lewej strony jest zawsze przenoszony na prawą. |
| | *** —(N)— | *Cewka wrażliwa na zbocze opadające*<br>(ang. *Negative transition-sensing coil*)<br>Skojarzona zmienna przyjmuje wartość *1* na czas jednego wykonania, jeśli połączenie z lewej strony zmieniło stan z *ON* na *OFF*. Stan połączenia z lewej strony jest zawsze przenoszony na prawą. |

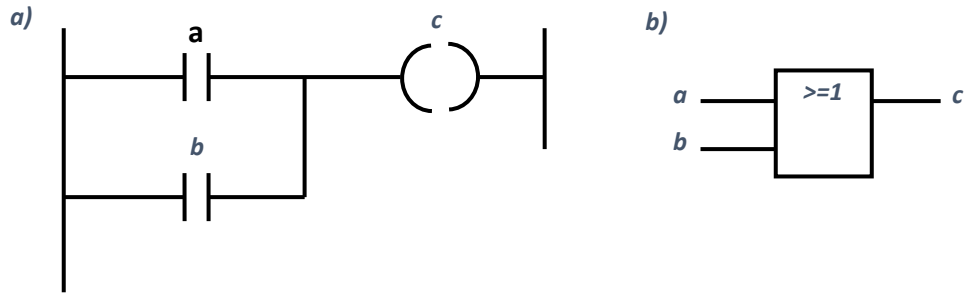
**Ad 5:** Norma IEC 1131-3 stanowi, aby w obrębie jednostki organizacyjnej programu, który został napisany językiem LD, wyznaczanie wartości wyjściowych sieci programowych powinno odbywać się w kolejności od góry do dołu, w miarę ich pojawiania się w „drabinkach" programu. Powinno to się odbywać z wyłączeniem przypadków, w których kolejność ta jest modyfikowana przez elementy sterowania wykonaniem, które zawarte są w Tabeli 23.

⇒ **Przykład programu dla sterownika PLC w języku LD (LAD)**




### 3.6. Język funkcjonalny FBD

Język **FBD**, który został określony w normie IEC 1131-3 jest drugim z języków graficznych, przeznaczonych do tworzenia programu dla sterownika PLC. Budowa, łączenie elementów oraz interpretacja programu, utworzonego w tym języku podlega zasadom, które zostały umówione w p.3.3.5., a które dotyczyły elementów wspólnych dla tworzenia programów w językach graficznych. Jednak istnieją pewne subtelne różnice między językiem **FBD** a językiem **LD**. Po pierwsze, wyjścia bloków funkcji w języku FBD nie powinny być zwierane ze sobą. W szczególności nie jest dopuszczalne stosowanie tzw. „sumy galwanicznej" (ang. ***Wired-OR***), co dopuszcza języka **LD**, a zamiast tego konieczność wykorzystania jawnej funkcji boolowskiej „**OR**" – rysunek 42.

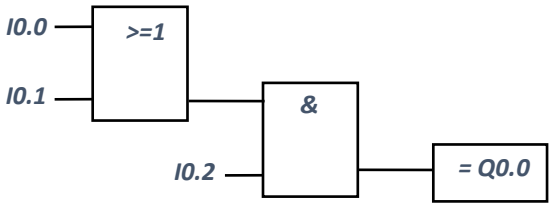



*Rysunek 42: Przykłady reprezentacji funkcji boolowskiej OR:*
*a) suma galwaniczna w języku LD; b) funkcja OR w języku FBD*

Po drugie, w obrębie jednostki organizacyjnej programu napisanego w języku FBD przy określaniu wartości wyjściowych sieci programowych obowiązuje zasada, że wyznaczanie wartości wyjściowej sieci powinno być zakończone przed rozpoczęciem wyznaczania wartości wyjściowej innej sieci, która wykorzystuje jeden lub więcej sygnałów wyjściowych tej pierwszej sieci.

⇒ **Przykład programu dla sterownika PLC w języku FBD**

Network 1




### 3.7. Graf SFC

Według normy IEC 1131-3 metoda **SFC** stanowi sposób tworzenia struktury programu w postaci schematu funkcji sekwencyjnej. Taki sposób tworzenia struktury programu pozwala na opisywanie zadań sterowania sekwencyjnego za pomocą grafów, które zawierają kroki (utożsamiane z etapami procesu sterowania) oraz warunki przejścia (tzw. tranzycje), czyli warunki logiczne, niezbędne do przemieszczania się sterowania od kroku poprzedniego do następnego.

Należy zaznaczyć, że w momencie powstania normy IEC 1131 graf **SFC** zaliczany był do struktur graficznych, za pomocą których można było jedynie modelować graficznie algorytmy sterowania procesem. Obecnie, na skutek rozwoju w zakresie oprogramowania narzędziowego do tworzenia programów dla sterowników PLC, graf **SFC** możemy zaliczyć do grupy języków graficznych programowania tych urządzeń.

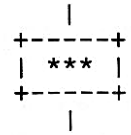
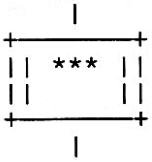
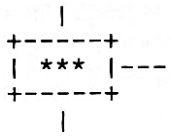
⇒ **Elementy grafu SFC**

Elementami, które są niezbędne do utworzenia grafu SFC są:

1. Krok sieci;
2. Tranzycja sieci (tzw. warunek przejścia);
3. Akcja;
4. Blok akcji z kwalifikatorami akcji.

**Ad 1:** Krok sieci algorytmu **SFC** określa etap procesu lub zestaw działań sterownika PLC skojarzonych z tym krokiem. Krok sieci oznaczany jest symbolem prostokąta narysowanego linią kreskową. Sposób przedstawiania kroków zgodnie z normą IEC 1131-3 ilustruje Tabela 26.

*Tabela 15: Sposób przedstawiania kroków zgodnie z normą IEC 1131-3*

| Nr | Reprezentacja | Opis |
|---|---|---|
| 1 |  | Krok – Forma graficzna z połączeniami bezpośrednimi<br>"***" = Nazwa kroku |
| |  | Krok początkowy – Forma graficzna z połączeniami bezpośrednimi<br>"***" = Nazwa kroku początkowego<br>(uwaga 2) |
| 2 | STEP *** :<br>(* Ciało kroku *)<br>END_STEP | Krok – Forma tekstowa bez połączeń bezpośrednich (patrz 2.6.3)<br>"***" = Nazwa kroku |
| | INITIAL_STEP *** :<br>(* Ciało kroku *)<br>END_STEP | Krok początkowy – Forma tekstowa bez połączeń bezpośrednich<br>(patrz 2.6.3)<br>"***" = Nazwa kroku początkowego |
| 3a | ***. X | Flaga kroku – Forma ogólna<br>"***" = Nazwa kroku<br>*** . X = Boolowska 1, gdy *** jest aktywny,<br>w przeciwnym razie Boolowskie 0 |
| 3b |  | Flaga kroku – Przyłączenie bezpośrednie zmiennej Boolowskiej<br>*** . X do prawej strony kroku "***" |
| 4 | ***. T | Czas upływający kroku – Forma ogólna<br>"***" = Nazwa kroku<br>*** . T = zmienna typu TIME<br>(patrz 2.6.2) |

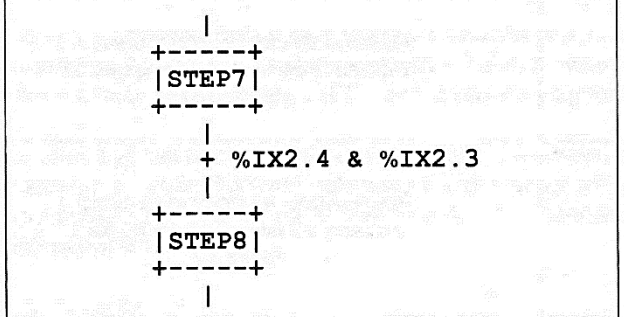
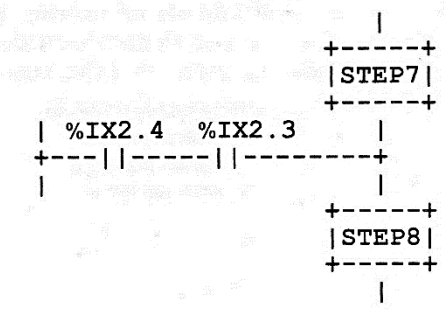
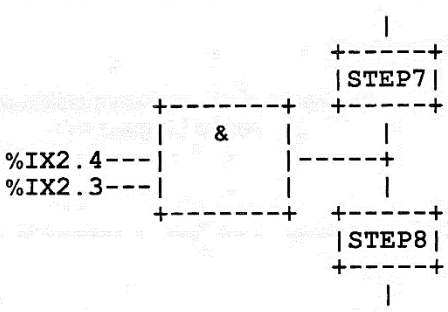
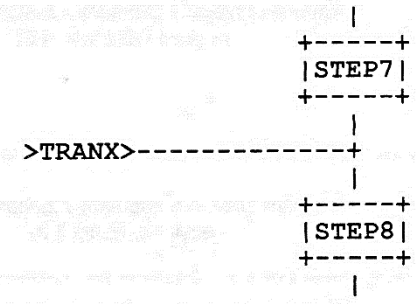
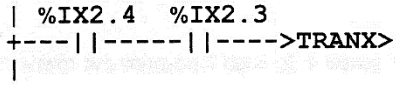
Krok sieci grafu **SFC** może być ***aktywny*** lub ***nieaktywny***. Określane jest to jego wskaźnikiem, który przybiera wartość zmiennej logicznej ***.X struktury boolowskiej, gdzie *** jest nazwą kroku. Gdy ta zmienna ma wartość logiczną „**1**”, to krok sieci jest aktywny, w przeciwnym przypadku krok jest nieaktywny.

Zgodnie z normą IEC 1131-3 stan początkowy (wyjściowy) procesu oraz sterownika PLC określa krok początkowy (inicjujący). W grafie **SFC** jest to wyróżnione graficznie jak w Tabeli 26, poz. 1b, czyli podwójnymi liniami jako krawędziami. Norma narzuca, aby dowolna sieć **SFC** miała tylko jeden krok początkowy.

Przy inicjowaniu systemu domyślnym początkowym czasem upływającym dla kroków sieci jest t#0s, domyślnym stanem początkowym dla zwykłych kroków jest boolowskie „0”, a dla kroków początkowych boolowskie „1”. (To ostanie wynika z faktu, iż musi istnieć warunek rozwoju algorytmu SFC, zatem po uruchomieniu programu (sterownika PLC) musi zaistnieć krok początkowy). Może jednak zaistnieć przypadek stanu początkowego grafu SFC inny od wyżej wymienionego. Taka sytuacja ma miejsce przy deklaracji tzw. podtrzymywania stanów bloków funkcji lub programu.

**Ad 2:** Tranzycja sieci(warunek przejścia) reprezentuje pojedynczy warunek lub warunki logiczne dla realizacji poszczególnych kroków programu, czyli etapów procesu. Spełnienie tego warunku przejścia jest podstawą do kontroli przechodzenia od jednego lub więcej kroków następujących po tej tranzycji, zgodnie z odpowiadającymi kierunkami łączenia w grafie **SFC**. Tabela 27 ilustruje modelowanie warunków przejścia w grafie **SFC**.

*Tabela 16: modelowanie przejść oraz warunków przejścia*

| Nr | Przykład | Opis |
|---|---|---|
| 1 |  | Krok poprzedni<br>Warunek przejścia zapisany z zastosowaniem języka ST (patrz 3.3)<br>Krok następny |
| 2 |  | Krok poprzedni<br>Warunek przejścia zapisany z zastosowaniem języka LD (patrz 4.2)<br>Krok następny |
| 3 |  | Krok poprzedni<br>Warunek przejścia zapisany z zastosowaniem języka FDB (patrz 4.3)<br>Krok następny |
| 4 |  | Użycie łącznika<br>Krok poprzedni<br>Łącznik przejścia<br>Krok następny |
| 4a |  | Warunek przejścia<br>Stosowanie języka LD<br>(patrz 4.2) |
| 4b |  | Stosowanie języka FBD<br>(patrz 4.3) |

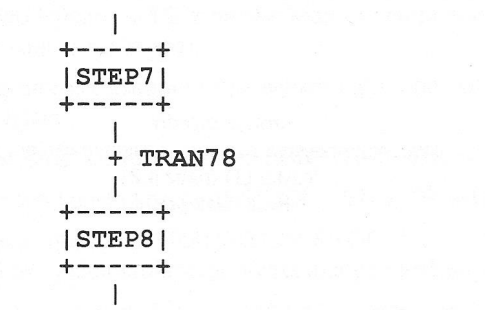
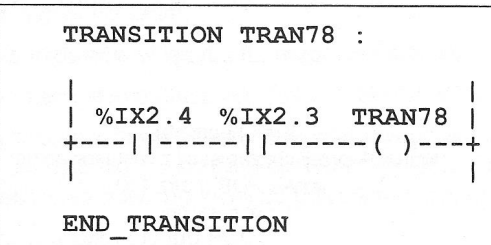
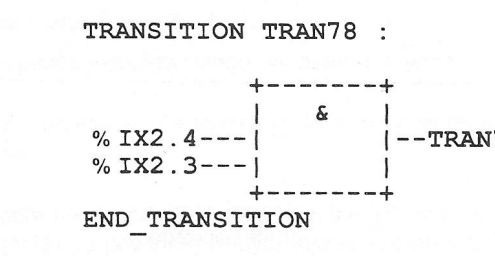
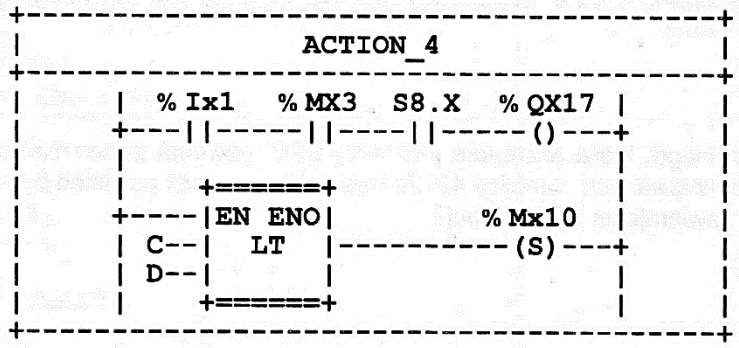
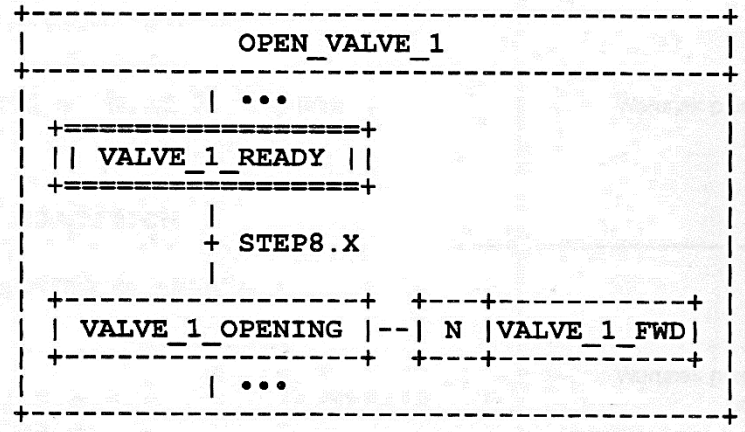
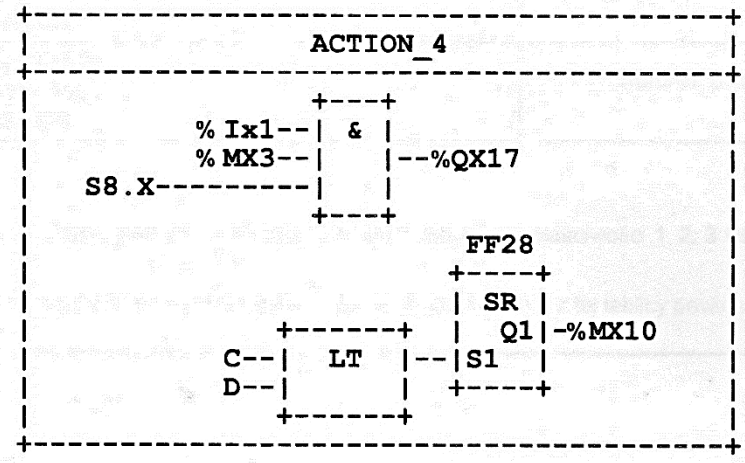
| Nr | Przykład | Opis |
|---|---|---|
| 5 | STEP STEP7 : END_STEP<br>TRANSITION FROM STEP7 TO STEP 8<br>:= %IX2.4 & %IX2.3 ;<br>END_TRANSITION<br>STEP STEP8 : END_STEP | Równoważnik tekstowy właściwości 1 zapisany z zastosowaniem języka ST (patrz 3.3) |
| 6 | STEP STEP7 : END_STEP<br>TRANSITION FROM STEP7 TO STEP 8 :<br>LD %IX2.4<br>AND %IX2.3<br>END_TRANSITION<br>STEP STEP8 : END_STEP | Równoważnik tekstowy właściwości 1 zapisany z zastosowaniem języka IL (patrz 3.2) |
| 7 |  | Użycie nazwy przejścia<br>Krok poprzedni<br>Nazwa przejścia<br>Krok następny |
| 7a |  | Warunek przejścia zapisany z zastosowaniem języka LD (patrz 4.2) |
| 7b |  | Warunek przejścia zapisany z zastosowaniem języka FDB (patrz 4.3) |
| 7c | TRANSITION TRAN78 :<br>LD %IX2.4<br>AND %IX2.3<br>END_TRANSITION | Warunek przejścia zapisany z zastosowaniem języka IL (patrz 3.2) |
| 7d | TRANSITION TRAN78 :<br>:= %IX2.4 & %IX2.3 ;<br>END_TRANSITION | Warunek przejścia zapisany z zastosowaniem języka ST (patrz 3.3) |

Tabela 27 pokazuje, że symbolem graficznym tranzycji jest poziomy odcinek prostopadły do pionowych kierunków łączenia. Każdą tranzycję opisuje określona zależność logiczna, która jest rezultatem modelowania logicznych zależności przyczynowo-skutkowych, określających warunki realizacji poszczególnych etapów procesu. Norma IEC 1131-3 dopuszcza, aby tranzycie mogły być modelowane w pozostałych językach programowania sterowników PLC, które obejmu te norma.

Norma IEC 1131-3 wskazuje, aby zakres nazwy tranzycji (warunku przejścia), użytej w grafie **SFC** powinien być lokalny w obrębie programowej jednostki organizacyjnej, w której dana tranzycja w grafie **SFC** została umieszczona. Jeżeli w czasie obliczania warunku przejścia pojawi się efekt uboczny, np. przypisanie wartości zmiennej nie będącej nazwą przejścia, to sytuacja taka powinna być uznana jako błąd.

**Ad 3:** Norma IEC 1131-3 stanowi, że z każdym krokiem sieci grafu SFC może być skojarzonych wiele akcji lub żadna. Norma przyjmuje, że krok, który nie posiada skojarzonych z nim akcji ma funkcję WAIT, tzn. oczekuje, aż warunek przejścia do kroku następnego będzie prawdziwy (ta tranzycja zostanie „zapalona"). Tabela 28 ilustruje przykłady deklaracji akcji w grafie **SFC**.

*Tabela 17: Przykłady deklaracji akcji w grafie SFC*

| | Przykład | Właściwości |
|---|---|---|
| 2l |  | Deklaracja graficzna zapisana w języku LD (patrz 4.2) |
| 2s |  | Włączenie w akcję elementów SFC |
| 2f |  | Deklaracja graficzna zapisana w języku FDB (patrz 4.3) |

| | Przykład | Właściwość |
|---|---|---|
| 3s | ACTION ACTION_4 :<br>%QX17 – %IX1 & %MX3 & S8.X ;<br>FF28 (S1 := (C<D)) ;<br>%MX10 := FF28.Q ;<br>END_ACTION | Deklaracja tekstowa<br>w języku ST<br>(patrz 3.3) |
| 3i | ACTION ACTION_4 ;<br>LD S8.X<br>AND %IX1<br>AND %MX3<br>ST %QX17<br>LD C<br>LT D<br>S1 FF28<br>LD FF28.Q<br>ST %MX10<br>END_ACTION | Deklaracja graficzna<br>w języku IL<br>(patrz 3.2) |

Norma stanowi, że akcją może być zmienna boolowska, zestaw rozkazów w języku **IL** (**STL**), zestaw instrukcji języka **ST**, zestaw drabinek w języku **LD** (**LAD**), zestaw sieci w języku **FBD** lub graf **SFC**.

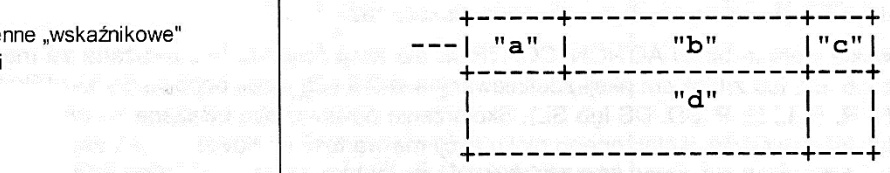
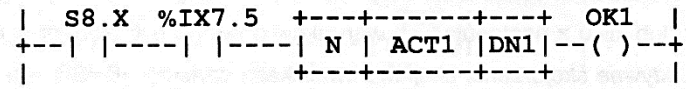
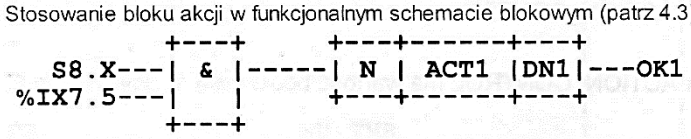
Akcje powinny towarzyszyć krokom za pomocą tekstowych ciał kroków lub graficznych bloków akcji (opisanych dalej). To kojarzenie akcji z krokami powinna zapewniać implementacja sterownika PLC, który realizuje graf SFC. Przykłady kojarzenia akcji z krokami pokazuje Tabela 29.

*Tabela 18: Przykłady kojarzenia akcji z krokami zgodnie z normą IEC 1131-3*

| Nr | Przykład | Właściwość |
|---|---|---|
| 1 | +----+ +-----+----------+---+<br>\| S8 \|--\| L \| ACTION_1 \|DN1\|<br>+----+ \|t#10s\| \| \|<br>\| +-----+----------+---+<br>+ DN1<br>\| | Blok akcji<br>(patrz 2.6.4.3) |
| 2 | \|<br>+----+ +-----+----------+---+<br>\| S8 \|--\| L \| ACTION_1 \|DN1\|<br>+----+ \|t#10s\| \| \|<br>\| +-----+----------+---+<br>+ DN1 \| P \| ACTION_2 \| \|<br>\| +-----+----------+---+<br>\| \| N \| ACTION_3 \| \|<br>\| +-----+----------+---+ | Konkatenacja bloków akcji |
| 3 | STEP S8 :<br>ACTION_1(L,t#10s, DN1) ;<br>ACTION_2(P) ;<br>ACTION_3(N) ;<br>END_STEP | Tekstowe ciało kroku |
| 4 | +-----+----------+---+<br>----\| N \| ACTION_4 \| \|---<br>+-----+----------+---+<br>\| %QX17 := %IX1 & %MX3 & S8.X ; \|<br>\| FF28 (S1 := (C<D)); \|<br>\| %MX10 := FF28.Q ; \|<br>+----------+ | Pole „d" bloku akcji<br>(patrz 2.6.4.3) |

**Ad 4:** Norma IEC 1131-3 stanowi, że blok akcji jest elementem graficznym grafu SFC, który łączy zmienną boolowską z jednym z kwalifikatorów akcji. (Kwalifikatory akcji podano w dalszej części tekstu). Połączenie to ma na celu stworzenie warunku zezwalającego dla akcji skojarzonej. Formalizm ten pokazano w Tabeli 30.

*Tabela 19: Właściwości bloku akcji zgodnie z normą IEC 1131-3*

<table>
<tr><th>Nr</th><th>Właściwość</th><th>Forma graficzna</th></tr>
<tr><td>1<br>2<br>3<br>4<br>5<br>6<br>7</td><td>"a" : Kwalifikator<br>"b" : Nazwa akcji<br>"c" : Boolowskie zmienne „wskaźnikowe"<br>"d" : Stosowanie akcji<br>język IL<br>język ST<br>język LD<br>język FBD</td><td>  </td></tr>
<tr><td></td><td colspan="2">Właściwość/Przykład</td></tr>
<tr><td>8</td><td colspan="2">Stosowanie bloku akcji w schemacie drabinkowym (patrz 4.2)<br>  </td></tr>
<tr><td>9</td><td colspan="2">Stosowanie bloku akcji w funkcjonalnym schemacie blokowym (patrz 4.3)<br>  </td></tr>
<tr><td colspan="3">UWAGI<br>1 Pole „a" może być pominięte, jeśli kwalifikatorem jest „N".<br>2 Pole „c" może być pominięte, jeśli nie używa się zmiennych wskaźnikowych.</td></tr>
</table>

Blok akcji daje możliwość opcjonalnego określenia boolowskich zmiennych wskaźnikowych, zaznaczonych za pomocą pola „c" w Tabeli 30, które mogą być ustawiane przez określoną akcję, by wskazywać jej zakończenie, przekroczenie czasu, powstanie błędu, itp. Jeżeli pole „c" nie istnieje, a pole „b" określa, że akcja powinna być zmienna boolowską, to zmienna ta, w razie potrzeby powinna być interpretowana jako zmienna „c".

Norma IEC 1131-3 stanowi, że z każdym skojarzeniem kroku z akcja lub z każdym wystąpieniem bloku akcji powinien być skojarzony tzw. kwalifikator akcji. Wartością tego kwalifikatora powinna być jedna z wartości wymienionych poniżej w Tabeli 31.

*Tabela 20: Kwalifikatory akcji zgodnie z normą IEC 1131-3*

| Kwalifikator | Opis |
|---|---|
| **brak** | Kwalifikator zerowy ( nie pamiętany) |
| **N** | Zapis - nie pamiętany |
| **R** | Kasowanie |
| **S** | Ustawianie |
| **L** | Ograniczenie czasowe |

| | |
|---|---|
| **D** | Opóźnienie czasowe |
| **P** | Pulsacja |
| **SD** | Zapis i opóźnienie czasowe |
| **DS** | Opóźnienie czasowe i zapis |
| **SL** | Zapis i ograniczenie czasowe |

⇒ **Reguły modelowania grafu SFC**

Przy modelowaniu algorytmu metodą *SFC* wymaga się spełnienia następujących zasad:

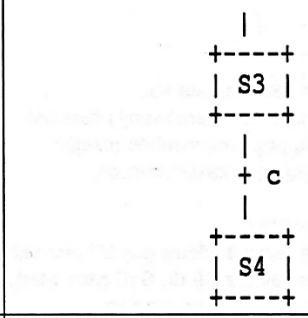
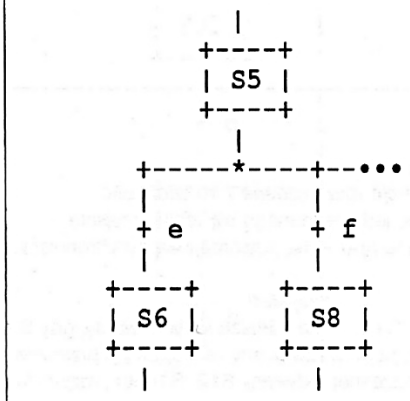
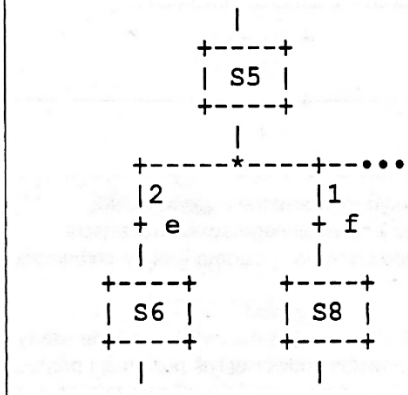
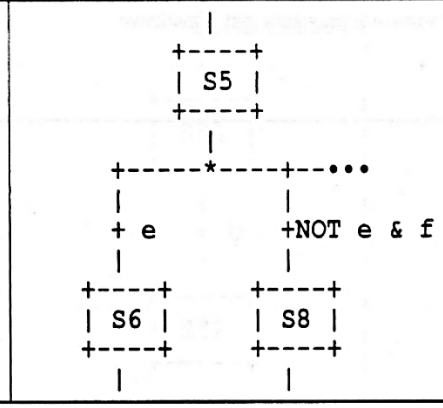
1. Krok początkowy reprezentuje proces w stanie początkowym. Krok początkowy jest w stanie aktywnym przy zainicjowaniu go przez użytkownika (np. przy sygnale z pulpitu sterującego) lub przez blok instrukcji.
2. Zmiana stanu procesu jest przedstawiana przez odblokowanie tranzycji przy spełnieniu następujących warunków:

- kroki poprzedzające tranzycję są aktywne;
- warunek logiczny określający tranzycję ma wartość „**1**”.

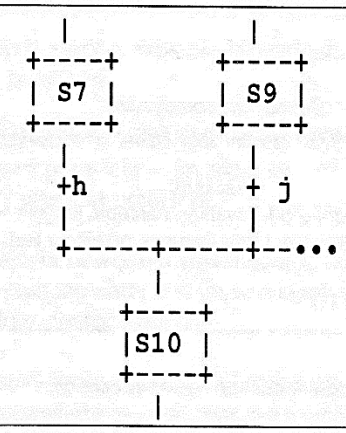
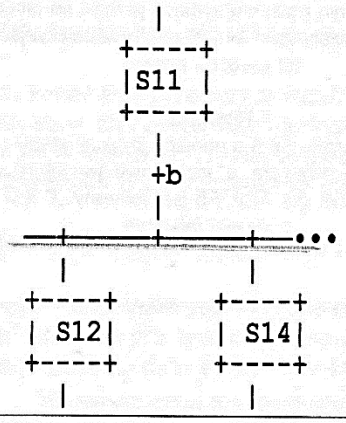
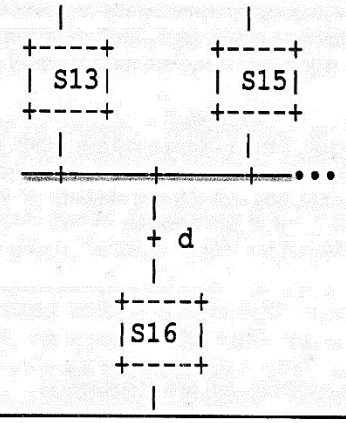
3. Tranzycja jest kasowana po spełnieniu określającego go warunku logicznego. Skasowanie tranzycji powoduje dezaktywację wszystkich kroków poprzedzających i aktywację wszystkich kroków bezpośrednio następujących po nim.
4. Bezpośrednie łączenie dwóch kroków/tranzycji jest zabronione. Kroki/tranzycje zawsze musi rozdzielać tranzycja/krok.
5. W celu modelowania procedur współbieżnych stosowana jest tranzycja specjalnego typu. Reprezentuje ona przechodzenie do jednoczesnej realizacji procedur sekwencyjnych lub jednoczesne zakończenie realizacji procedur sekwencyjnych. Ten typ tranzycji oznaczany jest linią podwójną.

Tabela 32 ilustruje zasady rozwoju kroków oraz stosowania tranzycji w sieciach **SFC**, stosowane przy modelowaniu algorytmów sterowania dla sterowników PLC.

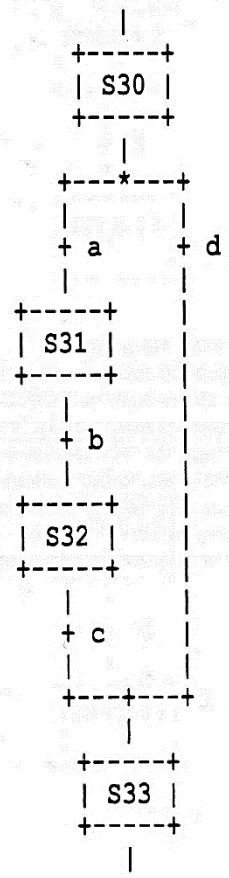
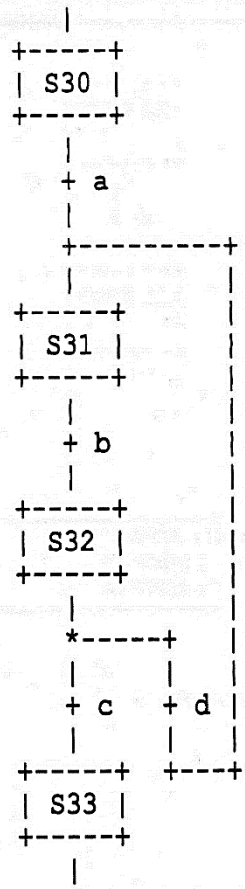
*Tabela 21: Zasady rozwoju kroków oraz stosowania tranzycji w sieciach SFC*

| Nr | Przykład | Zasada |
|---|---|---|
| 1 |  | *Sekwencja pojedyncza:*<br>Naprzemiennie występujący ciąg krok-przejście<br><br>*Przykład:*<br>Rozwój z kroku S3 do kroku S4 powinien nastąpić jedynie wtedy, gdy krok S3 jest w stanie aktywnym i gdy warunek przejścia jest prawdziwy. |
| 2a |  | *Rozbieżność lub wybór sekwencji:*<br>Wybór między szeregiem sekwencji jest reprezentowany przez tyle umieszczonych pod linią poziomą symboli przejść ile istnieje możliwych rozwojów. Gwiazdka wskazuje, że wartościowanie przejść ma priorytet od lewej do prawej.<br><br>*Przykład:*<br>Rozwój z kroku S5 do kroku S6 ma miejsce jedynie wtedy, gdy krok S5 jest aktywny i gdy warunek przejścia „e" jest prawdziwy lub rozwój z kroku S5 do S8 tylko wówczas, gdy krok S5 jest aktywny, „f" jest prawdziwe a „e" jest fałszywe. |
| 2b |  | *Rozbieżność lub wybór sekwencji:*<br>Gwiazdka, po której występują ponumerowane rozgałęzienia wskazuje, że priorytet wartościowania przejść jest określony przez użytkownika; rozgałęzienie z najniższym numerem ma najwyższy priorytet<br><br>*Przykład:*<br>Rozwój z kroku S5 do kroku S8 ma miejsce jedynie wtedy, gdy krok S5 jest aktywny i gdy warunek przejścia „f" jest prawdziwy lub rozwój z kroku S5 do S6 tylko wówczas, gdy krok S5 jest aktywny, „e" jest prawdziwe a „f" jest fałszywe. |
| 2c |  | *Rozbieżność lub wybór sekwencji:*<br>Połączenie rozgałęzień wskazuje, że użytkownik powinien zapewnić warunek wzajemnej rozłączności warunków przejść, zgodnie ze specyfikacją podaną w IEC 848.<br><br>*Przykład:*<br>S6 tylko wtedy, gdy S5 jest aktywny i warunek przejścia „e" jest prawdziwy, zaś z kroku S5 do S8 jedynie tylko wówczas, gdy S5 jest aktywny, „e" jest fałszywe a „f" jest prawdziwe. |

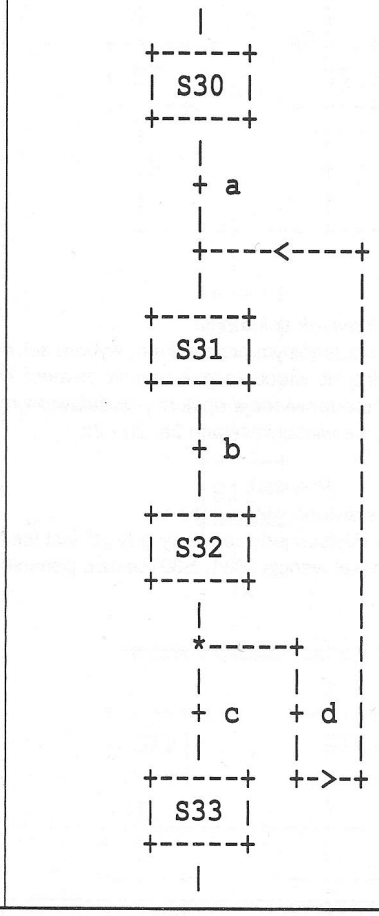
cd. Tabeli 32

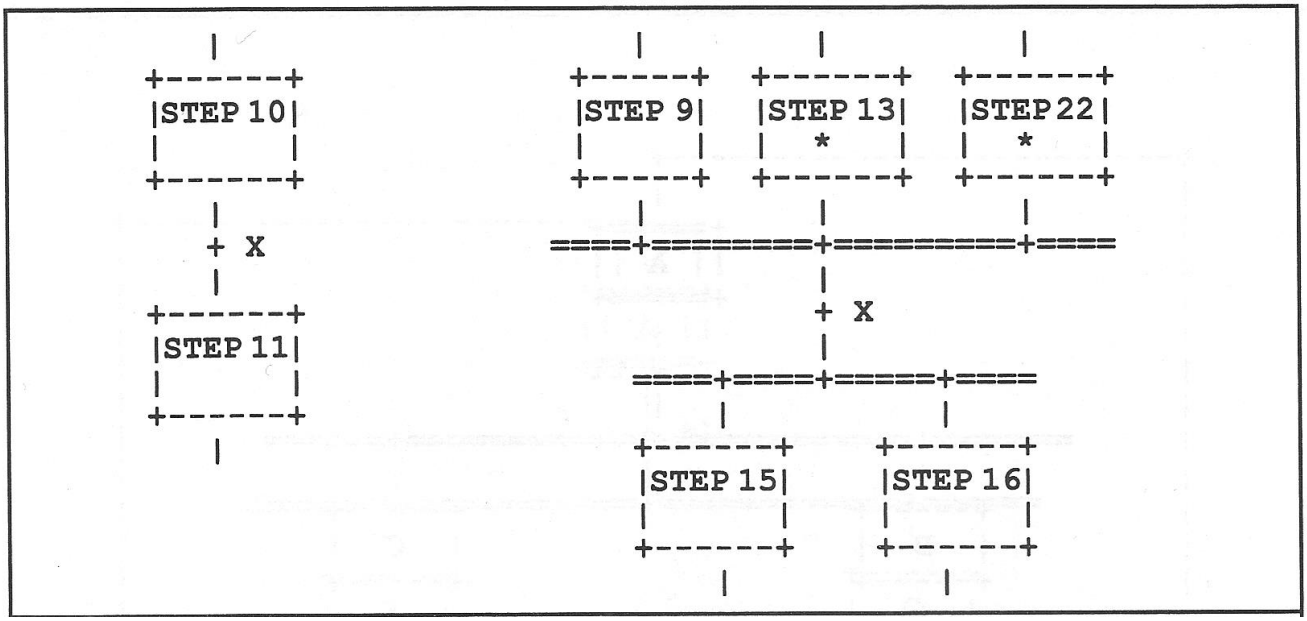
| Nr | Przykład | Zasada |
|---|---|---|
| 3 |  | *Zbieżność lub wybór sekwencji:*<br>Koniec wyboru sekwencji jest reprezentowany przez tyle umieszczonych nad linią poziomą symboli przejść, ile ścieżek wyboru ma być zakończonych.<br><br>*Przykład:*<br>Rozwój z S7 do S10 ma miejsce jedynie wtedy, gdy S7 jest aktywny i przejście „h" jest prawdziwe, albo z S9 do S10 tylko wtedy, gdy S9 jest aktywny i „j" jest prawdziwe. |
| 4 |  | *Sekwencje równoczesne – rozbieżność*<br>Można użyć jedynie jednego symbolu przejścia bezpośrednio *nad* podwójną poziomą linią synchronizacji.<br><br>*Przykład:*<br>Rozwój z S11 do S12, S14, ... ma miejsce jedynie wtedy, gdy S11 jest aktywny i warunek przejścia „b" związany ze wspólnym przejściem jest prawdziwy. Po równoczesnej aktywacji S12, S14 itd., rozwój każdej sekwencji przebiega niezależnie. |
| |  | *Sekwencje równoczesne – rozbieżność*<br>Można użyć jedynie jednego symbolu przejścia bezpośrednio *pod* podwójną poziomą linią synchronizacji.<br><br>*Przykład:*<br>Rozwój z S13, S15, ... do S16 ma miejsce jedynie wtedy, gdy wszystkie kroki powyżej podwójnej linii poziomej i przyłączone do niej są aktywne i gdy warunek przejścia „d" związany ze wspólnym warunkiem przejścia jest prawdziwy. |

cd. Tabeli 32

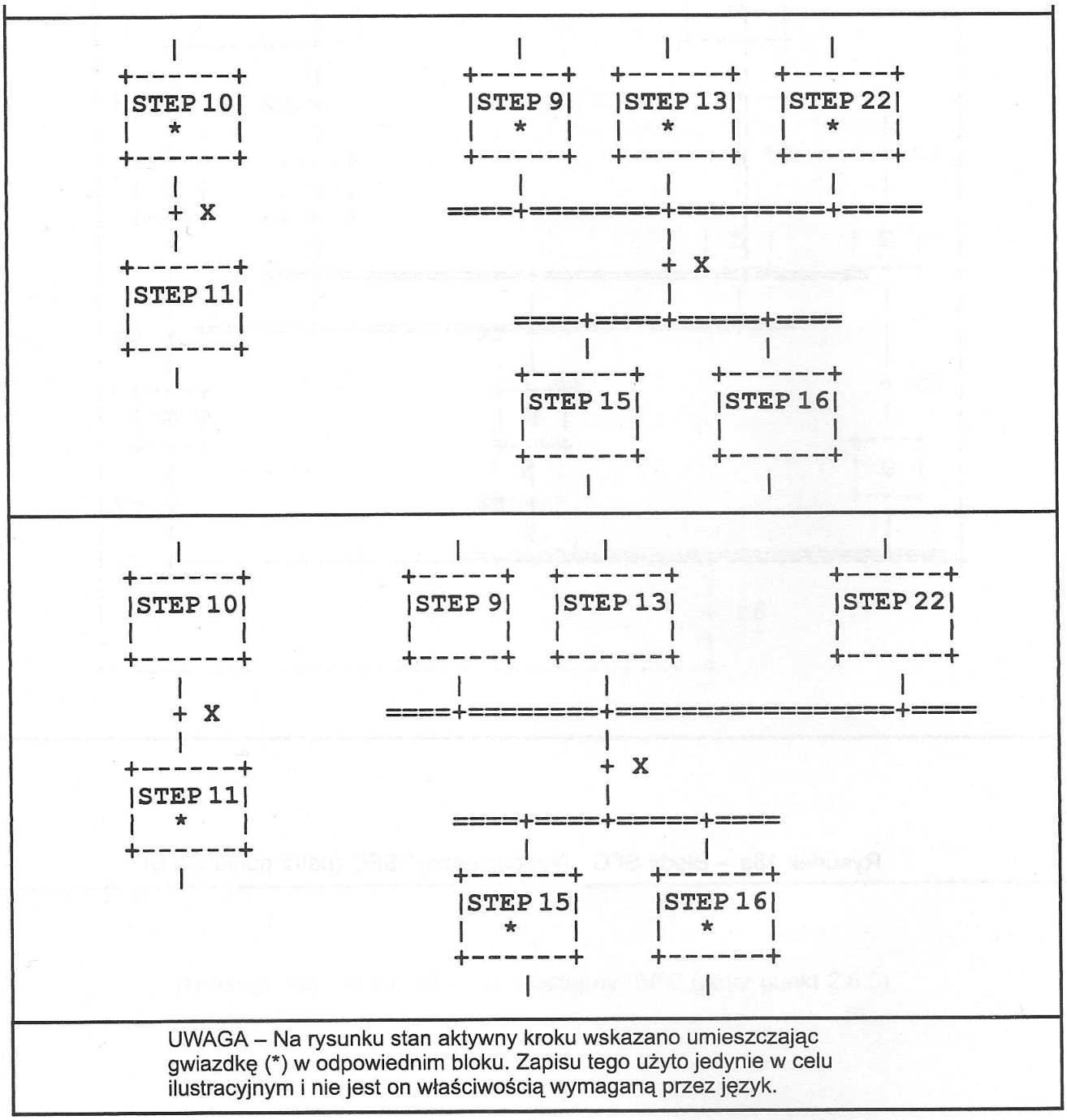
| Nr | Przykład | Zasada |
|---|---|---|
| 5a<br>5b<br>5c |  | *Przeskok sekwencji:*<br>„Przeskok sekwencji" jest szczególnym przypadkiem wyboru sekwencji (właściwość 2), w której jedno lub więcej rozgałęzień nie zawiera kroków.<br>Właściwości 5a, 5b i 5c odpowiadają opcjom przedstawionym, odpowiednio, we właściwościach 2a, 2b i 2c.<br><br>*Przykład:*<br>(przedstawiono właściwość 5a)<br>Rozwój z S30 do S33 ma miejsce jedynie wtedy, gdy „a" jest fałszywe i „d" jest prawdziwe, tzn. sekwencja (S31, S32) będzie pominięta. |
| 6a<br>6b<br>6c |  | *Pętla sekwencji:*<br>„Pętla sekwencji" jest szczególnym przypadkiem wyboru sekwencji (właściwość 2), w której jedno lub więcej rozgałęzień powraca do kroku poprzedniego.<br>Właściwości 6a, 6b i 6c odpowiadają opcjom przedstawionym, odpowiednio we właściwościach 2a, 2b i 2c.<br><br>*Przykład:*<br>(przedstawiono właściwość 6a)<br>Rozwój z S32 do S31 ma miejsce jedynie wtedy, gdy „c" jest fałszywe i „d" jest prawdziwe, tzn. sekwencja (S31, S32) będzie powtórzona. |

cd. Tabeli 32

| Nr | Przykład | Zasada |
|---|---|---|
| 7 |  | *Strzałki kierunkowe:*<br>Jeśli zachodzi konieczność lepszego zobrazowania sytuacji, to można użyć znaku "mniejszy niż" ( < ) ze zbioru znaków ISO/IEC 646 do wskazania przebiegu sterowania z prawa na lewo i znaku "większy niż" ( >) do reprezentacji przebiegu sterowania z lewa na prawo.<br>Jeśli wykorzystuje się tę właściwość, to odpowiedni znak powinien być umieszczony między dwoma znakami "–",<br>tzn. w sekwencji znaków "–<–" lub "–>–",<br>jak podano w załączonym przykładzie. |



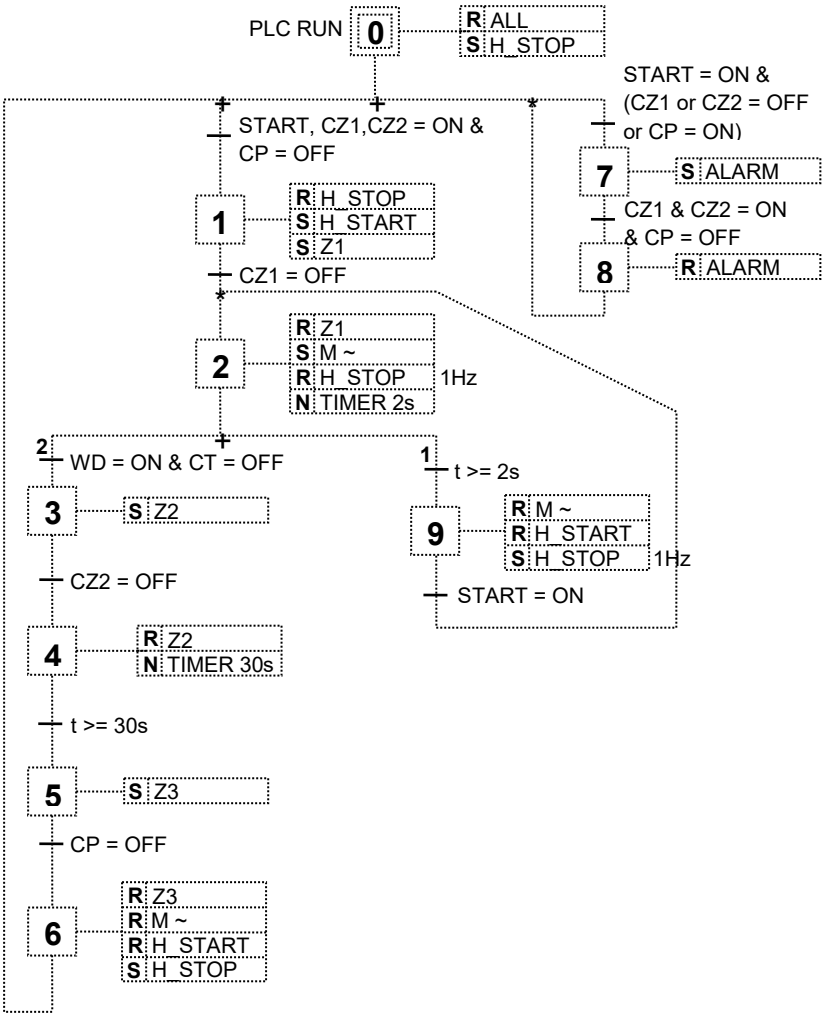
cd. Tabeli 32




UWAGA – Na rysunku stan aktywny kroku wskazano umieszczając gwiazdkę (*) w odpowiednim bloku. Zapisu tego użyto jedynie w celu ilustracyjnym i nie jest on właściwością wymaganą przez język.

⇒ **Przykład grafu SFC dla utworzenia programu dla sterownika PLC**

Rysunek 44 ilustruje przykład grafu SFC, na podstawie którego powinien być utworzony program użytkowy dla sterownika PLC. Przy tworzeniu programu użytkowego należy wykorzystać języki (metody) programowania PLC, ujęte w tej trzeciej części normy IEC 1131. (W momencie opracowania tej normy graf SFC uznano jako graficzny sposób ilustrowania działania algorytmu, który jest zarezerwowany dla systemów mechatronicznych. Obecnie graf SFC można zaliczyć do języków programowania PLC, gdyż istnieją środowiska programistyczne, które generują kod źródłowy programu użytkowego PLC bezpośrednio na podstawie grafu SFC. Przykładem takiego środowiska może być oprogramowanie narzędziowe ISAGrapf dla sterowników firmy PEP Modular Computers).




*Rysunek 44: Przykład grafu SFC dla utworzenia programu użytkowego PLC*

## 4. Podsumowanie

W rozdziale trzecim skupiono się na zilustrowaniu Czytelnikowi najważniejszych zagadnień, które objęły trzy rozdziały opracowanej w 1993 roku normy IEC 1131. Biorąc pod uwagę tworzenie programu użytkowego dla sterownika PLC najbardziej istotny jest właśnie ten trzeci rozdział tej normy. Zawiera on bowiem wytyczne dla producentów oprogramowania narzędziowego, które powinny służyć do opracowania środowisk narzędziowych do tworzenia programów użytkowych PLC. Rozdział ten zawiera zatem omówienie dwóch metod tekstowych tworzenia programu – język **IL** oraz **ST**, oraz trzy metody graficzne – **LD**, **FBD** oraz grafów **SFC**. Ta ostatnia metoda, jak już wspomniano wyżej stała się już narzędziem do tworzenia programu użytkowego PLC, zatem można ją zaliczyć do trzeciej metody (języka) graficznej tworzenia programu.

Autor zaznacza wyraźnie, że norma IEC 1131-3 pokazuje jedynie ogólne wytyczne, które dotyczą zasad tworzenia programów użytkowych dla sterowników PLC i tak te informacje powinny być przez Czytelnika postrzegane. Dopiero oprogramowanie narzędziowe dla konkretnego sterownika PLC zapewni finalne narzędzia do utworzenia „ładowalnego" programu PLC. Tak często używane przez autora pojęcia „program PLC" oraz „program użytkowy PLC" lub „program użytkownika" znaczą jedno i to samo – program, który powinien być załadowany do pamięci programu sterownika PLC.

# BIBLIOGRAFIA


1. Borelbach K.H., i inni: *Steuerungstechnik mit speicherprogrammierten steuerrungen SPS*. Munchen 1992.
2. Czemplik A., Jabłoński A.: *Stacje operatorskie w systemach automatyki - zadania i oprogramowanie*. IX Krajowa Konferencja Naukowo-Techniczna nt. Zadania mikroprocesorów w automatyce i pomiarach, Warszawa, Październik 1994.
3. Hajda J., Kasprzyk J., Wyrwał J.: *Programowanie sterowników PLC*. Gliwice 1998.
4. Jelonek K., Trawiński A., Zakrzewski D.: *Popularne standardy transmisji szeregowej. Przegląd interfejsów i protokołów komunikacyjnych*. Elektronizacja nr 6-8 1997.
5. Markiewicz H.: *Instalacje elektryczne*. Warszawa WNT 1996.
6. Mikulczyński T., Samsonowicz Z.: *Automatyzacja dyskretnych procesów produkcyjnych*. Warszawa WNT 1997.
7. Norma IEC 1131 Programmable Controllers. 1993.
8. Norma PN-89/M-42007/01 Automatyka i pomiary przemysłowe. Oznaczenia na schematach.
9. Norma PN-90/M-42007/02 Automatyka i pomiary przemysłowe. Oznaczenia funkcji systemów komputerowych.
10. OMRON, *Programmable Controllers*. Katalog 1995.
11. Sacha K., *Projektowanie oprogramowania systemów sterujących*. Wydawnictwo Politechniki Warszawskiej 1996.
12. Sacha K.: *Systemy czasu rzeczywistego*. Wydawnictwo Politechniki Warszawskiej 1997.
13. SCHIELE, *Programmable Controllers*. Katalog 1995.
14. Seta Z., *Wprowadzenie do zagadnień sterowania. Wykorzystanie programowalnych sterowników logicznych PLC.* Wydawnictwo MIKON, Warszawa 2002.
15. SIMATIC S5 - Step 5 Ladder 90. Manual. Siemens.
16. SIMATIC, *LAD/STL/FBD Programming Manual*. Siemens 1998.
17. SIMATIC - S7 300 *Programmable Controller, Instalation and Hardware. Manual*. Siemens 1998.
18. SIMATIC - S7 300 and M7 300, *Programmable Controllers, Module Specyfications, Reference Manual*. Siemens 1998.
19. SIMATIC - S7 200, *Programmable Controllers, Hardware and Instalation, Manual*. Siemens 1997.
20. Traczyk W.: *Układy cyfrowe automatyki*. Warszawa WNT 1974.
21. Winiecki W.: *Organizacja komputerowych systemów pomiarowych*. Wydawnictwo Politechniki Warszawskiej, Warszawa 1997.